Paradigms for Machine Learning

JEFFREY C. SCHLIMMER
School of Computer Science
Carnegie Mellon University, Pittsburgh, PA

PAT LANGLEY
AI Research Branch, Mail Stop 244–17
NASA Ames Research Center, Moffett Field, CA

NASA Ames Research Center
Artificial Intelligence Research Branch

Technical Report FIA–91–10

April 15, 1991

Paradigms for Machine Learning

JEFFREY C. SCHLIMMER

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

PAT LANGLEY

AI Research Branch (MS 244–17)
NASA Ames Research Center
Moffett Field, CA 94035

Abstract

In this paper we describe five paradigms for machine learning – connectionist (neural network) methods, genetic algorithms and classifier systems, empirical methods for inducing rules and decision trees, analytic learning methods, and case-based approaches. We consider some dimensions along which these paradigms vary in their approach to learning, and then review the basic methods used within each framework, together with open research issues. We will argue that the similarities among the paradigms are more important than their differences, and that future work should attempt to bridge the existing boundaries. Finally, we discuss some recent developments in the field of machine learning, and speculate on their impact for both research and applications.

To appear in S. Shapiro (Ed.), *Encyclopedia of artificial intelligence* (2nd ed.). New York: John Wiley & Sons.

1. Machine Learning and Artificial Intelligence

One central insight of artificial intelligence is that expert performance requires domain-specific knowledge. Machine learning is the subfield of AI that studies the automated acquisition of such knowledge. The aim is intelligent systems that learn — i.e., that improve their performance as the result of experience. Researchers study learning for a variety of reasons: to discover general principles of intelligence, to better understand human learning, and to automate the process of knowledge acquisition. However, the discipline is united in its concern with mechanisms for learning, and methods proposed with one goal in mind often serve other purposes equally well.

Despite its separate identity, there are two important senses in which machine learning is an integral part of the larger AI field. First, learning researchers must consider central AI issues of knowledge representation, memory organization, and performance. Second, learning can occur in any domain requiring intelligence, whether the basic task involves classification, problem solving, reasoning, natural language processing, or vision. Thus, one can view machine learning as more a framework for AI research and development than as a subfield of AI.

In the following pages, we review the current state of machine learning. We begin by identifying relevant dimensions along which learning systems differ. Next we describe five different *paradigms* for machine learning that have emerged over the past two decades. These include:

- connectionist (neural network) learning methods;
- genetic algorithms and classifier systems;
- empirical methods for inducing rules and decision trees;
- analytic learning methods; and
- case-based approaches to learning.

Despite differences in the representations and algorithms used by these approaches, all aim to improve performance through the acquisition of knowledge from experience. Moreover, we will see that they must address many similar issues, and that they share many open problems. After examining each of these paradigms, we consider some recent developments in the field as a whole and their potential impact on future research and applications.

One related area that we will not examine is *knowledge acquisition* (e.g., Marcus, 1988). Like machine learning, this subfield of AI is concerned with improving performance through the storage of domain-specific knowledge. However, research on knowledge acquisition focuses primarily on direct interaction with an expert by asking questions, taking advice, and the like. In contrast, most work on machine learning emphasizes the acquisition of knowledge through experience with an external environment or experience with internal problem-solving traces.

2. Dimensions of Machine Learning Research

Before delving into recent research on machine learning, let us identify some dimensions along which alternative learning paradigms vary. Below we discuss six issues — the representation of experience, the representation of acquired knowledge, the performance task, the difference between supervised and unsupervised learning, the distinction between incremental and nonincremental learning, and the complementary notions of induction and explanation.

2.1 Representation of Experience

Most learning is based on experience, and this requires a representation for the experiential input given to the learning system. Researchers have employed three broad classes of data structures, each with different representational power. The simplest describes each individual experience as a list of binary *features*, each corresponding to the presence or absence of some aspect of the environment. For example, a particular symptom (say jaundice) may be present or absent for a given patient. Connectionist and genetic algorithms typically employ feature-based input.

A second scheme assumes a known set of *attributes*, each having a set of mutually exclusive values. Thus, one might describe an object as blue or red, but not both at the same time. Some attributes take on numeric values (e.g., length, weight). Attribute-value representations are typically used by empirical methods, such as those for rule induction and decision-tree construction.

A final approach employs relational or structural representations. These describe relations between two or more objects, such as the fact that object A is above object B (i.e., arbitrary predicates). Such structural information can be stated in many formalisms, including predicate logic and semantic networks. Although relational schemes have significantly more expressive power than the other representations, they also introduce significant complexity into the matching process, and this affects learning methods that use them. Research on analytic learning typically deals with relational data structures, as does some work on empirical rule learning and case-based reasoning.

2.2 Representation of Acquired Knowledge

All learning systems acquire new knowledge, and they must represent that knowledge in some fashion. The choice of knowledge representation is central in that it strongly influences choices of performance and learning components. In fact, the five learning paradigms we discuss in Section 3 below are distinguished largely in terms of representational issues.

One choice is whether to store individual, concrete experiences or only abstractions based on these data. Most work on machine learning has taken the latter path, and attempts to move beyond the data to build more general knowledge structures that summarize previous experience. Of course, hybrid approaches are possible; these are often called *case-based* approaches, and we discuss them in Section 3.5. Note that methods that attempt to store only concrete experiences must at some point generalize beyond the data if they hope to apply to new experiences. Thus, the real issue is whether such abstraction occurs aggressively as new experiences are stored (as in most approaches) or lazily as needed during knowledge access (as with case-based approaches).

For abstract knowledge structures, another choice is whether to use a logical, discrete formalism or one that involves numeric, continuous information. This is primarily a choice between using logical connectives or numeric ones, respectively. Empirical rule learning and analytic learning methods have predominantly used the first path, whereas connectionist systems have relied on the second. Genetic algorithms can be profitably viewed as combining these two formalisms, as can recent work on probabilistic learning. A related choice is whether to use a aggregated, coarse-grained representation or a finer-grained one. These options are sometimes referred to as "symbolic" and "subsymbolic" representations, respectively. Conventional wisdom associates the former scheme

with empirical and analytic methods, and associates the latter with connectionist and genetic approaches. However, as we argue in Section 3.6, these associations are more myth than fact.

2.3 The Performance Task

Learning involves improvement in performance, and thus learning cannot occur in the absence of some performance task. The vast majority of machine learning research has focused on two broad classes of domains — *classification* and *problem solving*. Each has led to different concerns, distinct insights, and unique applications.

Intelligent agents are repeatedly confronted with the need to classify or label their experience. For example, upon encountering certain symptoms, a doctor diagnoses a specific disease. The generic task can be easily stated: given some description of an experience, along with a set of known classes and their descriptions, assign that experience to one or more classes. The most frequent successes in expert systems have involved just such classification or diagnostic tasks (Davis & Lenat, 1982). Much of the work on machine learning has focused on learning descriptions for classification.

However, an intelligent agent must also be able to solve novel problems and formulate plans. For example, when going to a meeting, an agent devises a path from its current location to the target site. The generic task can be stated as: given some desired state or goal, find some sequence of actions that takes you from the current state to the desired one. Navigation and robot manipulation are the most obvious applications of problem-solving techniques, but there are many others, including scheduling. This area has been less popular within machine learning, but much of the recent work has focused on acquiring knowledge to improve the speed and quality of methods for planning and reasoning.

Finally, intelligent agents also participate in other high-level behaviors that may not fit neatly into either the classification or problem-solving paradigms. Such behaviors include the design of new artifacts, communication with other agents, and the control of motor effectors. To date, the machine learning community has placed little emphasis on these areas, but because of their potential, this will undoubtedly change in the future.

2.4 Supervised and Unsupervised Learning

Another dimension that influences learning is the degree of supervision. In some cases, a tutor or domain expert may be present to give the learner immediate feedback about the appropriateness of its behavior. This situation is typically called *supervised* learning. In other cases, an *unsupervised* learner may have to fend for itself. Here the learner has little or no external guidance in building knowledge structures or composing solutions to problems; at most, the environment provides coarse-grained feedback about the learner's overall effectiveness on the task at hand.

Early work in machine learning focused on the simpler supervised task, and it continues to receive considerable attention. There are good practical reasons for this interest. Although experts generally have poor ability to introspect about their domain knowledge, they are much better at providing examples of correct and incorrect behavior.

Both forms of learning can occur in many contexts, though they take on different forms in different domains. In classification domains, the supervised task is usually called *learning from examples*, and the individual training experiences are called *instances*. This task can be stated as:

- *Given*: A set of instances (e.g., patient symptoms) and their associated classes (e.g., diseases);
- *Find*: A general description for each class that matches only its instances.

This basic task has been examined within all the major paradigms of machine learning, and until recently, the vast majority of research papers have dealt with this topic. However, there is now also considerable interest in unsupervised concept learning, in which the learner must decide for itself not only the class in which it should place each experience, but also the number of such classes. This has been called *clustering* (Fisher & Langley, 1985).

In problem-solving domains, supervised learning occurs when a tutor is available to suggest the correct operator or subgoal at each point in the search for a solution. Methods that operate in this context are sometimes called *learning apprentices*, and they show promise as a way of extracting knowledge from an expert (Mitchell, Mahadevan, & Steinberg, 1985) in problem-solving and design domains. However, more research in this area has focused on unsupervised learning, in which the agent must distinguish desirable actions from undesirable ones for itself. This has been called the *credit assignment* problem (Sleeman, Langley, & Mitchell, 1982), and researchers have explored a variety of responses across a number of paradigms.

2.5 Incremental and Nonincremental Learning

Some learning algorithms process experiences one at a time, whereas others process a large set of experiences at once. The former class is often called *incremental* and the latter *nonincremental*. However, examining only the surface behavior of a system can be misleading, since an inherently nonincremental system can always be run in incremental mode and vice versa. For example, if one lets a nonincremental technique retain exact copies of previous experience in memory, it can process the first experience, then process the first two experiences, then process the first three experiences, and so forth. Similarly, one can take an incremental method and iteratively run it through the same set of experiences again and again. A better definition of *incremental* involves the number of previously seen experiences one must reprocess during learning (Schlimmer & Fisher, 1986).

Both nonincremental and incremental approaches have their advantages. The former can collect statistics about all the data, giving a learning method more information on which to base its decisions. On the other hand, incremental methods tend to be more efficient and, in principle, can deal with much larger data sets. Much of the recent machine learning research has focused on incremental approaches, though there are some notable exceptions in the literature on connectionist methods and on the induction of rules and decision trees.

2.6 Inductive and Analytic Learning

A sixth important dimension relates to the type of learning used to acquire knowledge. In one class of approaches, *inductive* learning methods formulate knowledge based mainly upon observed data. Empirical, genetic, and connectionist techniques are instances of this general strategy. These

methods are inductive in the sense that they move beyond their input, and generalize it to create knowledge that was not previously in memory. In contrast, *analytic* learning methods formulate knowledge based mainly on other knowledge already in memory. Explanation-based learning is a clear example of this type of approach; it uses prior knowledge to explain new experiences, then simplifies the explanation and stores it in memory.

Inductive methods have also been called *knowledge-level learners*, since their acquired knowledge structures change the deductive closure of the system (Dietterich, 1987). In contrast, analytic methods are *symbol-level* learners; their compiled knowledge may increase efficiency but leave the deductive closure unchanged. However, later we will consider some applications in which inductive techniques produce symbol-level learning, and consider some conditions under which analytic methods can produce knowledge-level shifts.

Of course, more unified approaches are possible. A learner could use prior knowledge from the domain to constrain induction, producing partial explanations that still require inductive leaps to form general structures. Alternatively, instead of using deductive rules to construct explanations, a learner could use plausible inference rules; the resulting explanations may be plausible, but they are not guaranteed to hold, and thus involve a sophisticated form of induction. Some researchers have argued that the unification of inductive and analytic approaches should be given high priority (Langley, 1989).

3. Five Paradigms for Machine Learning

Now we can examine some recent research on machine learning. We have divided the field into five "paradigms" based upon the basic representations and learning methods employed — connectionist approaches, genetic algorithms, empirical rule learning, analytic learning, and case-based methods. In each case, we review the basic approach, examine recent advances, and consider some open research issues.

Each paradigm is loosely defined not only by a shared set of assumptions, concerns, and methods, but also by the amount of interaction among researchers. Generally, people within a given paradigm read and reference each other's papers, attend the same meetings and presentations, and all too often ignore work in other paradigms. In Section 3.6, we identify some important relations among these paradigms that reveal more similarities than appear at first glance.

3.1 Connectionist (Neural Network) Learning

Some of the earliest research on machine learning focused on connectionist methods (Nilsson, 1965), and recently there has been a resurgence of interest in this approach. The name derives from the basic representation for learned knowledge — a network composed of nodes connected by directed, weighted links (sometimes called "neural" networks because of the suggestive similarity between the computational style of network nodes and neurons). Such systems typically assume that inputs are represented as a set of binary features, with each feature being present or absent. In operation, features that are present activate the network's initial nodes. Then, the weights on links from these

nodes to others determine whether subsequent nodes will be activated. The process iterates until activation has a chance to reach the network's final nodes, with the output of the network being the activation of the final nodes.

Let us examine the connectionist framework's position on the dimensions discussed in Section 2. Learning consists of modifying link weights to better mimic the desired relations among the inputs and outputs. Thus connectionist approaches lend themselves naturally to performance tasks that involve classification, but they can be adapted to more complex domains. Although not logical in form, the connections encode the acquired knowledge and summarize the data encountered. Predominantly, researchers have focused on methods for supervised learning situations, but there are some exceptions. Connectionist researchers have also examined many alternative strategies for adjusting link weights; many are incremental and all are inductive in nature.

3.1.1 PERCEPTRONS AND LINEAR THRESHOLD UNITS

The simplest form of connectionist network is the *perceptron* or *linear threshold unit* (Rosenblatt, 1962). In this framework, there is a single output node to which each input node is connected by a single weighted link. The output node also has an associated threshold. Given a datum, the output node sums the weights of the links from active input nodes (those whose features are present). If this sum exceeds the threshold, the output node is activated; otherwise it remains inactive. Figure 1 presents an example of a simple perceptron.

Despite their simplicity, perceptrons can represent a variety of functions. For example, consider a network in which N links have a weight of one and all others are zero. If the threshold is set to N, the network encodes a "rule" that matches only when the *conjunction* of the nonzero features are present. Similarly, if the threshold is set to one, the same network encodes a *disjunctive* "rule." Using a threshold of K (where $N > K > 1$) lets the network concisely express the K of N function. Unlike the former two, this latter function is difficult to represent using logical notation. Moreover, allowing weights other than one supports an even broader class of functions. In fact, given the appropriate weights, a perceptron can represent any *linearly-separable* class. In other words, if we view the F features as defining an F-dimensional space, the network can describe any class that involves placing a single hyperplane between the instances of two classes.

There are a number of straightforward methods for learning appropriate link weights given example input-output pairs. One of these, the perceptron learning rule, learns only when it makes a prediction error. If the output unit is not active when it should be, then its incoming weights are too low; accordingly, they are incremented by a small constant. By the same reasoning, if the output unit is active when it should not be, the weights are too high and are decremented by the same small constant. Part of the appeal of this incremental method is that it is guaranteed to converge on any linearly-separable class given a finite number of instances (Minsky & Papert, 1969).

Other learning methods, such the LMS procedure (Widrow & Hoff, 1960), modify each weight differentially in an attempt to reduce the mean-squared error between the desired and generated output. This approach can be run incrementally or nonincrementally (using all available instances),

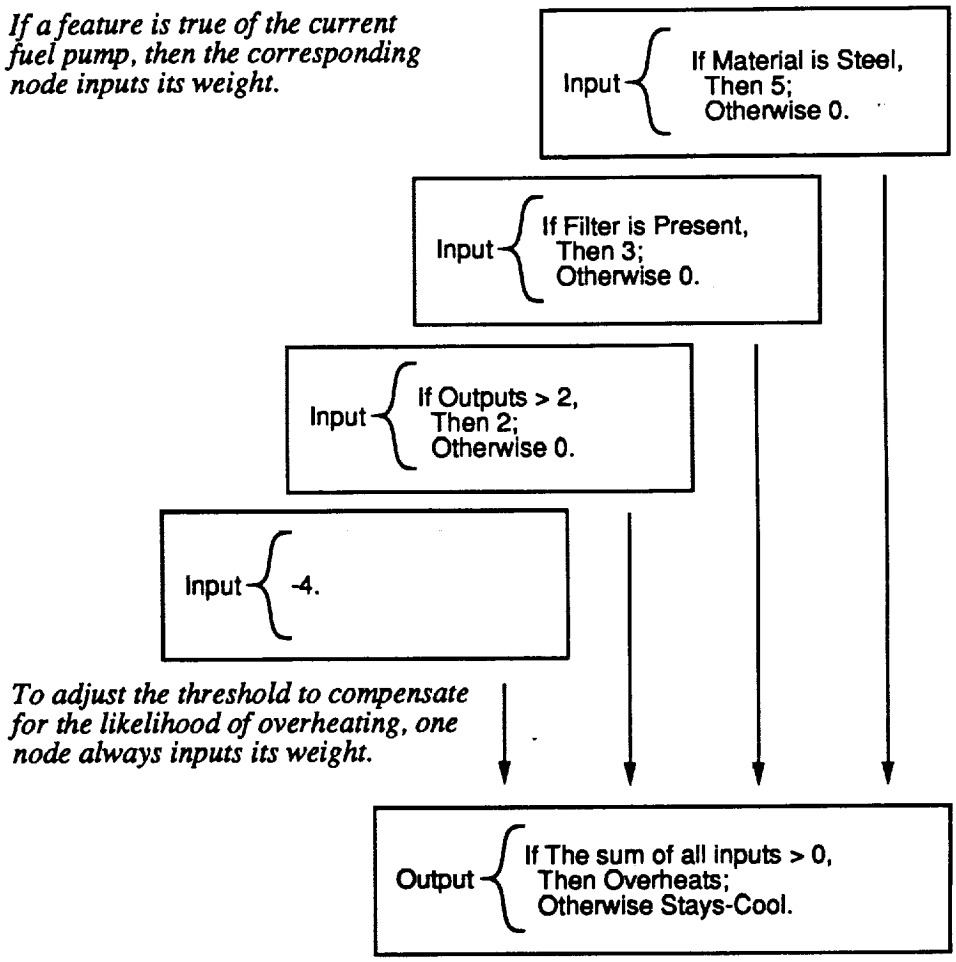

Figure 1. Schematic of the organization of a linear threshold unit for a simple domain in which one must predict whether a fuel pump will overheat based on the material, the number of outputs, and the presence of a filter. Inputs correspond to instance features (shown at the top of the figure). If these features are present, their weights are summed by the output unit (shown at the bottom). Learning modifies the weights associated with inputs and thereby alters the summed output. An additional constant input (lowest and leftmost input) lets the linear threshold unit adjust its threshold.

and it can be generalized to handle networks with continuous rather than binary inputs. Linear threshold units have been used to learn regular and irregular endings for the past tense of English verbs (Rumelhart & McClelland, 1986), and a perceptron-like learning method has demonstrated its ability to acquire expert-level performance in the game of checkers (Samuel, 1959).

3.1.2 BACKPROPAGATION IN MULTILAYER NETWORKS

Despite their attractiveness, single linear threshold units cannot represent or learn class descriptions that are not linearly separable. Although higher-order components like quadratic threshold nodes might be feasible, a more common approach uses a network with multiple layers. These structures include intermediate, or *hidden*, nodes that are indirectly connected to the inputs and outputs of the network. Given sufficient breadth in this structure, the network can express an arbitrary

function of the input features. Moreover, if the structure is relatively shallow (i.e., only short paths from inputs to outputs), large networks of this sort are computationally efficient to use. However, learning the weights for such networks is another matter entirely, and much of the recent research on connectionist learning has focused on this issue.

One of the most intuitive approaches to learning appropriate link weights in a multiple layer network applies the LMS procedure recursively. Known as *backpropagation*, this procedure first propagates activation through the network in the normal, forward direction. Based on the differences between observed and desired outputs, backpropagation uses LMS to compute the desired activation levels on the hidden nodes one level back. Not only does this indicate the appropriate weight change in the final links, it also allows backpropagation to treat the hidden nodes as if they were output nodes. Using the difference between the observed and desired activation for hidden nodes, backpropagation applies LMS recursively until it reaches the input nodes.

Like the perceptron learning rule, backpropagation is conducting a search in the space of link weights. Effectively, both carry out a hill-climbing search in which the gradient is defined by error reduction. Unlike the convergence result for the simpler learning rule, backpropagation may become stuck in local optima. This has not emerged as a significant problem in studies to date, but it remains an open issue. More pressing problems include a very slow rate of learning and some dependence on the number of hidden nodes. One noteworthy application demonstrates that backpropagation can acquire pronunciation knowledge that accurately predicts phonemes from English text (Sejnowski & Rosenberg, 1987).

3.1.3 Alternative Approaches to Connectionist Learning

Researchers have explored a number of other approaches to learning in multi-layer networks. For example, Boltzmann machines are a probabilistic technique based on an analogy with thermodynamics, in which nodes settle into stable configurations as the "temperature" of the system decreases (Ackley, Hinton, & Sejnowski, 1987). Such methods must be run many times, in order to reach "equilibrium" and to collect statistics about the probability of connected nodes being active simultaneously. As a consequence, they are typically even slower than backpropagation. Nevertheless, Boltzmann machines also have advantages, and active research continues in the area, including applications to speech recognition (Prager, Harrison, & Fallside, 1986).

Other researchers have taken a reinforcement learning approach. In this framework, instead of fine-grained feedback about each of the network's outputs, the only information available is a single evaluation score for the network's overall behavior on each instance. For example, Barto, Sutton, and Anderson's (1983) AR-P algorithm rewards or penalizes each weight in the network equally as a function of the reinforcement evoked by the network's overall behavior. This learning scheme has been successfully applied to a number of domains, including a pole-balancing task involving the dynamic control of forces over time. Like the other methods, this approach extends to domains in which the inputs are real-valued rather than binary.

In other studies, researchers have explored the behavior and capabilities of cyclic and deeply nested network structures. For example, a cyclic network has its input nodes connected to its

output nodes and vice versa. This design allows networks to memorize patterns, and given a partial or noisy pattern to recall, the net can reconstruct the complete, noise-free original (Kohonen, Oja, & Lehtiö, 1981). In a deeply nested network, hidden nodes are ordered, and each successive hidden node receives input from all prior hidden nodes. Nesting the hidden nodes capitalizes on the representations they have learned, and the resulting structure tends to generalize better (Fahlman & Lebiere, 1990).

3.1.4 OPEN ISSUES IN CONNECTIONIST LEARNING

Research on learning in the connectionist framework has made significant strides since the early results with perceptrons, and initial applications have started to emerge. Still, a number of serious issues remain to be addressed:

- *Increasing the rate of learning.* Existing methods learn very slowly, often requiring many iterations through the training instances. Future research should examine the factors that affect the learning rate and develop connectionist methods that learn more rapidly (Fahlman, 1988; Hampson & Volper, 1987).
- *Generalization.* Current methods converge on weight settings that summarize training instances, but sometimes this is at the expense of accurate generalization over unseen instances. This issue becomes especially important when the training data are noisy (Fisher & McKusick, 1989; Knight, 1989; Weiss & Kapouleas, 1989).
- *Structural knowledge.* Some domains seem inherently relational, but connectionist methods rely on feature-based representations. The framework must be adapted to represent and learn from relational and structural input (Hinton, 1986).
- *Sequential behavior.* Connectionist techniques lend themselves to parallel implementations, but they have difficulty carrying out ordered actions like those required for problem solving. Extended architectures are needed that can handle sequential behavior, along with learning methods that can support them (Elman, 1990; Mozer & Bachrach, in press).
- *Incorporating domain-specific bias.* Perceptrons have a strong bias towards learning linearly separable classes, but multi-layer networks search a much larger space. Future research should examine methods for incorporating biases into multi-layer networks that constrain the learning methods' search and improve their learning rates (Towell, Shavlik, & Noordewier, 1990).

Growing numbers of AI researchers are examining the connectionist paradigm seriously and many are concerned with issues of learning. Theoretical analyses and experimental studies have begun to reveal a deeper understanding of these methods' advantages and drawbacks, pointing the way to the extensions and improvements outlined above.

3.2 Genetic Algorithms and Classifier Systems

Genetic algorithms are a family of adaptive search methods that derive their name from a loose analogy with genetic change in a population of individuals. Like connectionist methods, most genetic algorithms assume a feature-based representation of instances and events. However, rather

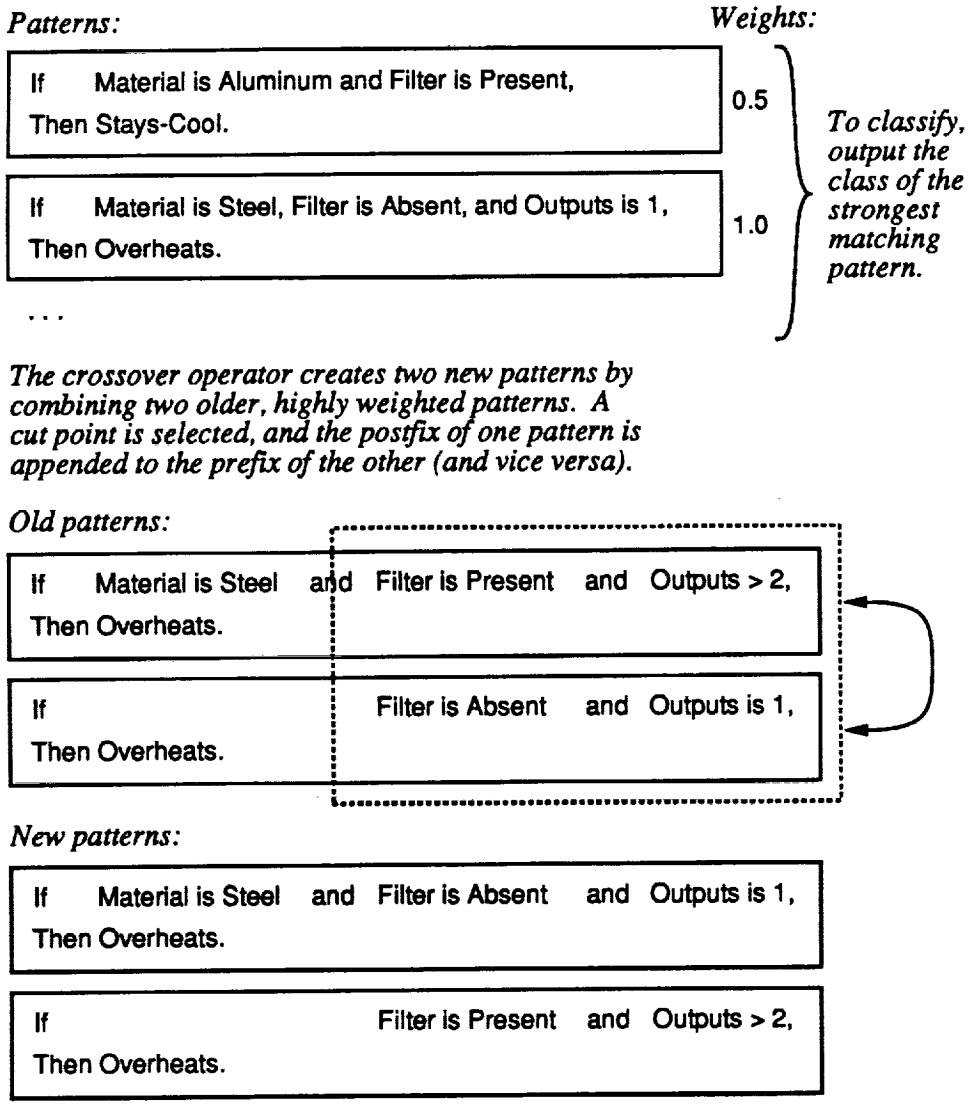

Figure 2. Schematic of the representations and operators used in the genetic algorithm as applied to the fuel-pump domain. The weights of conjunctive patterns (shown at the top of the figure) are used to classify instances. Learning modifies both the weights of patterns and their conditions. The simplest operator generates new patterns that are minor random variations of old ones (not shown). The crossover operator (shown at the bottom) splices together an arbitrary prefix of one pattern with an arbitrary suffix of another (and vice versa).

than using a weighted network to represent acquired knowledge, they employ a disjunctive set of knowledge structures or *patterns*. Each pattern is conjunctive and specifies the presence (or absence) of some features. Patterns also have an associated weight, sometimes called the pattern's *fitness*, that summarizes its performance on past experiences. Given a new instance, a stochastic scheme follows the recommendations of strong matching patterns to reach a decision (Wilson, 1987).

In terms of the dimensions of Section 2, this paradigm lends itself naturally to performance tasks that involve both classification and problem solving, as we discuss below. Research on genetic

algorithms has focused on supervised learning situations, although often the feedback is some overall score, as in the reinforcement learning framework discussed above. Researchers have explored a number of inductive approaches to incrementally modifying the content of individual patterns and their weights, and they have successfully applied genetic algorithms and classifier systems to a variety of tasks (Goldberg, 1990).

3.2.1 LEARNING WITH GENETIC ALGORITHMS

In addition to using a different learned representation, genetic algorithms and connectionist techniques also carry out a different type of search during learning. Whereas connectionist methods apply a hill-climbing operator to a single state (one set of weights), genetic algorithms apply several heuristic operators to a set of states (patterns). As shown in Figure 2, genetic algorithms follow three steps in processing new experiences: updating pattern strengths, applying search operators, and pruning ineffective patterns. Methods for updating weights vary widely between specific genetic algorithms, and we discuss several below. For the second step, genetic algorithms typically use two operators, *crossover* and *mutation*, that apply to strongly weighted patterns to produce syntactically similar new patterns; crossover is analogous to gene splicing, whereas mutation introduces random, minor variations. To maintain the size of the pattern set, the third step replaces prior, weakly weighted patterns with new patterns. In a sense, patterns compete with each other to produce "offspring" in the next cycle or "generation." There is a large body of both theoretical and empirical evidence showing that, even for very large and complex search spaces, genetic algorithms can rapidly locate effective knowledge structures using about 50 to 100 patterns.

Applying genetic algorithms to classification tasks is straightforward when instances can be represented as features. The set of knowledge structures is initialized to N random patterns, some of which will be very specific and others of which will be quite general. Each pattern is assigned a class. To adjust the weights, each time a pattern is successful (i.e., it matches an instance from its own class or it fails to match an instance from another class), its weight in incremented; each time a pattern is unsuccessful (i.e., fails to match its class or matches another class), its weight is decremented. Because pattern weights are decremented when they fail to match instances in their class, if the concept to be learned is disjunctive, the weight of every pattern will be decremented at one time or another. Nevertheless, pattern strength is still a useful heuristic for directing search through the space of descriptions, and genetic algorithms can learn complex disjunctive concepts, even in noisy domains (Wilson, 1987). This ability results in part from the inclusion of many patterns in each generation, some of which compete and others of which complement each other, in that they come to occupy different 'niches'.

3.2.2 CLASSIFIER SYSTEMS AND PROBLEM SOLVING

Classifier systems are an architecture for problem solving that incorporates a genetic algorithm as a component. Most work along these lines assumes a simple, forward-chaining production system, consisting of a set of condition-action rules and a dynamic working memory. Each rule is weighted and has one or more feature patterns as its condition and a single pattern as its action. Memory

contains a set of fully specified patterns, or *messages*, that have a value for all features. Through encoding conventions, some messages originate as inputs to the system and others trigger outputs.

The classifier system framework extends the standard forward-chaining recognize-act cycle. On every cycle, each rule whose conditions match messages in memory makes a *bid* proportional to its weight. One or more rules are then selected for application, with probabilities proportional to their bids. The selected rules are applied by adding the patterns in their actions to working memory. These new messages may allow other rules to match, and the cycle continues.

There are two main aspects to learning in classifier systems. First, a genetic algorithm is used to generate new candidate rules from existing, strongly weighted rules (similar to the process described above). Second, the classifier system adjusts the weights of rules based on their contribution to desirable behavior. This involves assigning credit to useful rules and blame to faulty ones. An effective approach to this latter learning problem is called the *bucket brigade* (Holland, 1985). As rules apply, they pass along a portion of their weight to the rules that applied in the cycle before them. Some rule ultimately is rewarded directly by the environment, and this reward is iteratively passed back through the rules in the application chain, increasing their weights. In the simplest case, the weight of any rule that participates in the chain eventually converges on the amount received by the last rule in the chain. As a consequence, rules that apply but do not lead to external reward are consistently paying out their weight without receiving any reinforcement, and their weight diminishes. Classifier systems have been successfully applied to a variety of domains, including regulation of gas flow through pipelines (Goldberg, 1985) and survival in a resource-scarce environment (Booker, 1988).

3.2.3 Alternative Uses of Genetic Algorithms

Unlike classifier systems, which apply operators to propose individual new rules, another approach to using genetic algorithms for problem solving applies genetic operators to entire rule sets. Rather than exploring variations of rule *conditions*, the operators primarily explore variations on rule *combinations*. Because of this, the knowledge structure is composed of multiple rule sets, each of which constitutes a forward-chaining production system that has an associated weight.

In this framework, the weight of each rule set is evaluated by running the rules on a set of training problems. Since rule sets with strong weights tend to be selected by the search operators, useful combinations of rules are propagated through the knowledge structure, and less useful rule combinations are gradually eliminated. The operators occasionally introduce new rules, but these are always evaluated in the context of their rule set. The power of these ideas has been demonstrated by a state-of-the-art poker-playing system (Smith, 1983) and by an effective system for multiple class discrimination in the domain of human gait analysis (Schaffer & Grefenstette, 1985). Recent research also shows that a combination of this approach and classifier systems performs better in some domains than either in isolation (Grefenstette, 1988).

3.2.4 OPEN ISSUES IN GENETIC ALGORITHMS

Although considerable progress has occurred in the understanding of genetic algorithms since their inception, many open research issues remain. These include:

- *Alternative representations.* Genetic algorithms typically assume a feature-based representation of knowledge. Future work should explore the application of these methods to more sophisticated representations (Gordon & Grefenstette, 1990; Koza, 1989), adding new genetic operators if necessary.
- *Acquired representations.* Classifier systems can represent complex class descriptions by the organization, variability, and distribution of weights in clusters of rules. However, we need to better understand the underlying nature of such learned representations and how are they acquired (Belew & Forrest, 1988).
- *Emergence of useful symbols.* Classifier systems can use *tags* to build associations between rules, producing behavioral sequences. Recent work has explored the development of such internal symbols and the conditions under which they emerge (Shaefer, 1987).
- *Credit assignment.* The issue of credit assignment is central to applying genetic algorithms to problem solving, and a variety of methods have been proposed, including weight update, conflict resolution, and the use of "taxes." Research is needed to determine the conditions under which each approach behaves well, and to explore hybrids that might do better than any method in isolation (Grefenstette, Ramsey, & Schultz, 1990).
- *Incorporating domain knowledge.* Genetic algorithms seem especially well suited for knowledge-lean domains in which extensive search is necessary, but they may also be able to use and refine existing domain knowledge.
- *Population size.* In some domains, genetic algorithms should behave significantly better than hill-climbing techniques (e.g., connectionist methods). Some research has studied the effect of the number of patterns in a knowledge structure (Robertson, 1988), but future work should identify the broader conditions under which maintaining a redundant knowledge structure is worth the cost.

Researchers in the genetic algorithm community are already attacking these problems, but more work remains before this promising approach achieves its full potential.

3.3 Empirical Learning Methods

Another community of machine learning researchers have studied *empirical* methods for the acquisition of more explicit knowledge structures. Let us consider the dimensions of Section 2 for this approach. Like connectionist and genetic methods, these techniques are inductive, in that they move beyond training instances to make predictions about novel cases. Unlike them, empirical learning methods have employed relational and structural representations for both experiences and acquired knowledge, though the majority of research has thus far focused on propositional representations. Much of the early work on empirical learning dealt with classification domains, but there has also been progress on problem solving and natural language acquisition.

Currently there are a variety of well-understood methods that learn production rules, decision trees, and concept hierarchies (to name a few of the many learned representations). Some methods require the close supervision of a tutor, as described in Section 2.4, whereas others learn in an unsupervised fashion. Some techniques require all instances at the outset, whereas others learn incrementally and can process new instances with little additional effort. In this section we review three main approaches to empirical learning, along with some open research problems.

3.3.1 EMPIRICALLY LEARNING PRODUCTION RULES

One common scheme for representing domain expertise uses production rules whose conditions test properties of experiences and whose actions specify classifications. Researchers have explored a variety of empirical methods for learning such rules from a set of preclassified training instances, and most approaches rely on the fact that the space of rule conditions are partially ordered according to generality. Thus, one can start with the most specific possible description, using a generalization operator to remove or relax conditions. Alternatively, one can start with the most general possible description, using a specialization operator to add or constrain conditions. The candidate elimination algorithm (Mitchell, 1977) employs both of these ideas to carry out a bidirectional exhaustive search to identify conditions for classification rules. For each possible class, the algorithm maintains a *version space* that summarizes the space of hypothesized conditions in terms of a most specific boundary set and a most general boundary set. New positive instance may indicate the need for more general descriptions, forcing revision of the specific boundary, whereas negative instances may suggest more specific descriptions, leading to revision of the general boundary. This continues until the algorithm converges on a single conjunctive description in both sets, or until one of the boundary sets becomes empty, indicating an inconsistency.

The candidate elimination algorithm assumes that a single, conjunctive rule can describe each class, and that training instances are free of noise. However, disjuncts and noise are common in applied settings. Another appealing family of learning methods relaxes these assumptions and uses heuristic search to limit computational expense. These methods employ beam search or related methods to find individual rules that discriminate between positive and negative instances of a class. Search may occur from general to specific rules or in the opposite direction. During each search, candidate rule conditions are minimally specialized (or generalized) in all possible ways, each specialization (or generalization) is heuristically evaluated for predictive accuracy on the training instances, and the best are further modified. Search terminates when none of the new specializations (or generalizations) are statistically better predictors than their predecessors, at which point the best candidate is used to construct a classification rule. To handle disjunctive domains, some methods then remove all positive instances from the training set that are covered by this rule and repeat the search process over the remaining instances, continuing until all positive instances are covered by some rule (Michalski, 1983; Clark & Niblett, 1989). Figure 3 depicts this approach graphically.

Methods of this sort have been successfully applied to moderately realistic tasks. For example, in the domain of lymphography, some rule-learning systems (Michalski, 1987; Clark & Niblett, 1989) have equaled the classification accuracy of human experts (82% correct). Another rule-learning

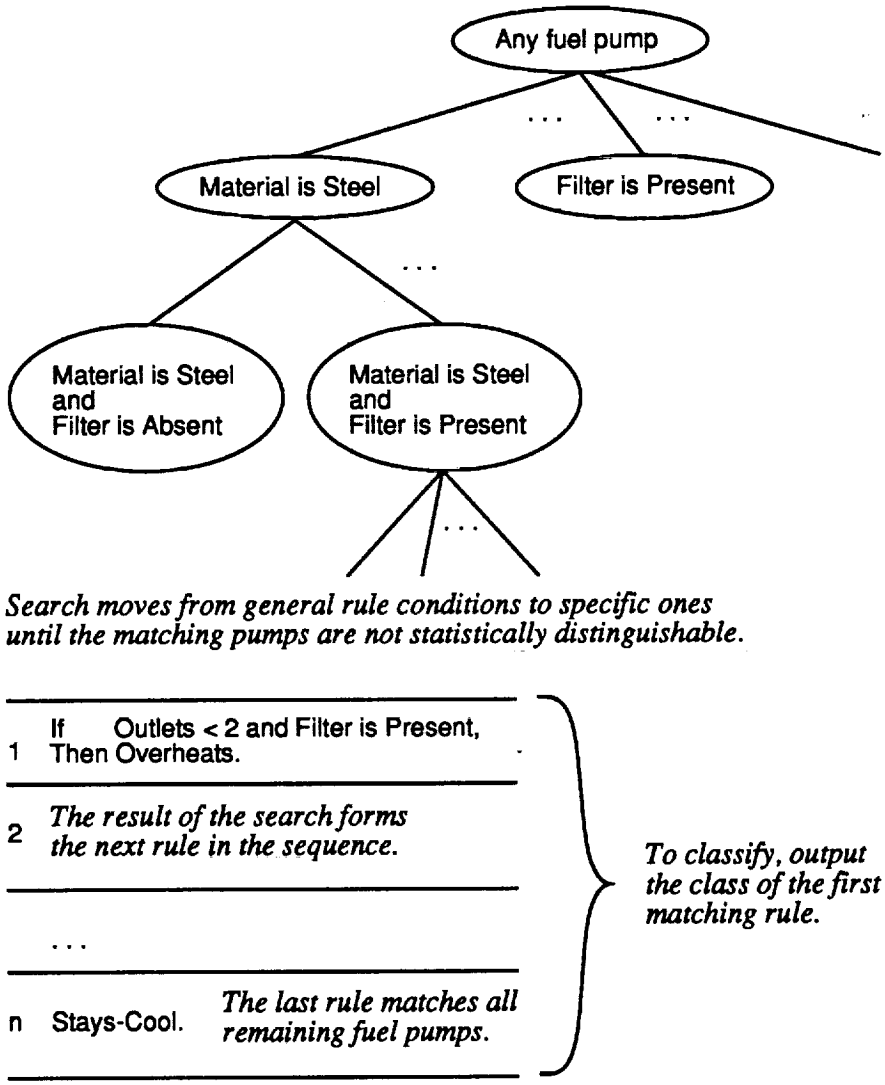

Figure 3. Schematic of the representation and search used in CN2, a recent rule learning method (Clark & Niblett, 1989), in the fuel-pump domain. Learned rules (shown at the bottom of the figure) are ordered, with the last rule identifying the class of any remaining instances. Search maintains a limited size boundary as it looks for rule conditions that are both predictive and reliable. Initially, it looks for a consistent description of a single class given all the instances, and subsequently it searches for a consistent description of those examples not covered by any rules found thus far. Search terminates when all instances are covered by some rule.

system (Schlimmer, 1987) has demonstrated similar results for recognizing poisonous mushrooms (95%) and predicting the political party of Congressmen based on their voting records (90%).

In spirit, methods for learning production rules are similar to connectionist and genetic methods. However, there are some significant differences. As we have seen, most of these methods take advantage of the fact that the space of class descriptions can be partially ordered by generality. Thus, most rule-learning schemes search using specialization operators, generalization operators, or both. Second, the search in most rule-learning methods is biased toward finding simple descriptions.

If this is appropriate in a given domain, then these methods tend to learn much more quickly than connectionist or genetic methods. Finally, some rule-learning methods represent experience and learned knowledge using relational and structural representations, giving them more expressive power than other approaches.

Although methods for empirical rule learning were originally designed with classification domains in mind, they can also be applied to problem-solving tasks. For example, given a set of legal operators for carrying out state-space search, the same methods can acquire the *heuristic* conditions under which each operator *should* be applied. However, this approach requires first identifying appropriate and inappropriate applications of each operator, and this is equivalent to assigning credit and blame to steps along a search path. One response to this latter issue involves waiting until a complete solution has been found. Then, steps along the solution are labeled as appropriate operator applications, whereas all steps leading off the solution are labeled as inappropriate (Langley, 1985; Mitchell, Utgoff, & Banerji, 1983). Another response to the credit/blame assignment issue involves interacting directly with a domain expert who provides immediate feedback about the desirability of each action. In either case, one can then apply empirical learning methods to the appropriate and inappropriate applications (as positive and negative instances-instances of the class, respectively), producing heuristic rules as output.

3.3.2 CONSTRUCTING DECISION TREES

Other work on empirical learning takes quite a different approach to the supervised learning task. This framework assumes the same input as systems that learn production rules (i.e., a set of instances assigned to classes), but the learned knowledge is represented as a *decision tree* (Brieman, Friedman, Olshen, & Stone, 1984; Quinlan, 1983). Each nonterminal node of this tree specifies some attribute to test, each branch specifies an alternative value, and each terminal node specifies a class. To classify a new instance, a decision tree iteratively tests non-terminal node attributes of that instance and follows matching branches until it reaches a terminal leaf that classifies the instance.

The most common decision-tree learning method uses a divide-and-conquer algorithm, selecting domain attributes to partition the instances and recursively building sub-decision trees to describe partitions. An evaluation function selects the most discriminating attribute for each non-terminal node's test. Instances are partitioned based on their value for the test attribute, and subtrees are constructed to describe each partition. As the process iterates, subtrees are complete when all instances in their partition have the same class (or when there are no more attributes to test). This process can be viewed as a greedy, general-to-specific search through the space of decision trees. Figure 4 shows an example of this approach.

Decision-tree methods have been applied to a variety classification tasks, representing a mix of synthetic and natural domains. For example, induced decision trees can successfully recognize lost chess endgames (100% for losses in three-ply) (Quinlan, 1983), and they can accurately classify thyroid diseases (99%) (Quinlan, Compton, Horn, & Lazarus, 1986). The latter application resulted in a decision tree that outperformed a hand-crafted expert system that took years to construct.

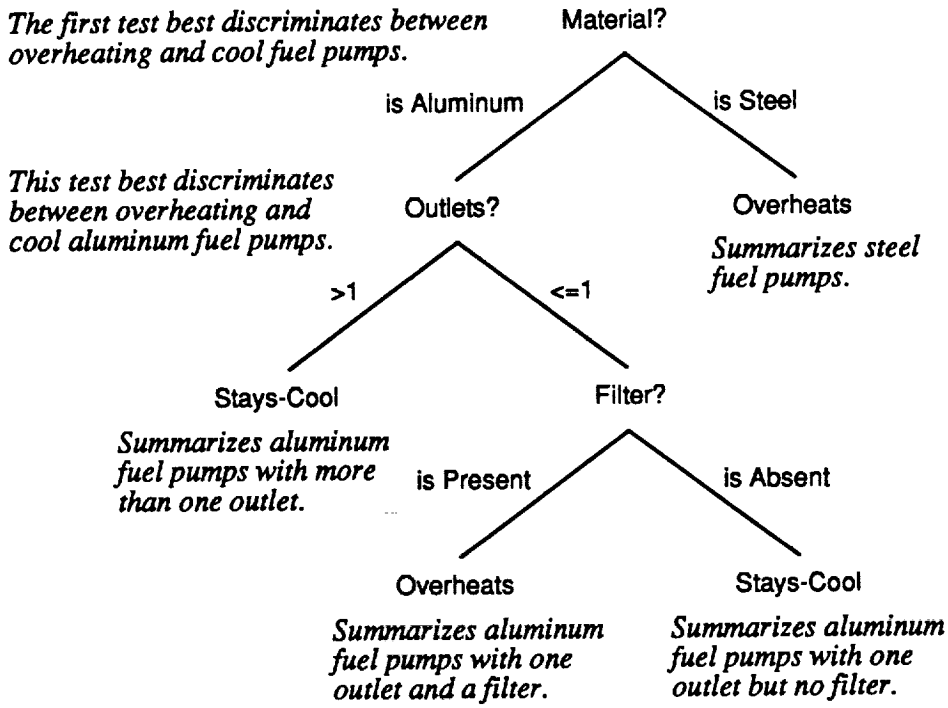

Figure 4. Schematic of the representation and search used in decision tree learning for the fuel-pump domain. To classify an instance, the test at the root of the tree (shown at the top of the figure) is applied first, and based on the outcome, classification proceeds with the appropriate subtree. The search begins by determining the single most discriminating test. Instances are then partitioned by their outcome for this test. The standard method terminates search when all instances in a partition are of the same class (or there are no more tests), but some variants use statistical tests to halt earlier as an effort to avoid overfitting.

Methods for inducing decision trees constitute some of the most widely studied algorithms within the machine learning community. Extensions include techniques for pruning trees in response to noisy training data (Brieman et al., 1984; Kononenko, Bratko, & Roskar, 1984; Quinlan, 1986b), methods for detecting useful thresholds on numeric attributes (Quinlan, 1986a), and algorithms for incrementally revising a tree in response to new data (Schlimmer & Fisher, 1986; Utgoff, 1989). Decision-tree methods have received significant attention within applied AI circles, and some industrial groups have used them to automatically construct expert-level diagnostic systems.

3.3.3 FORMING CONCEPT HIERARCHIES

In unsupervised concept learning tasks, no expert is available to classify instances for the learner. Instead, the learner is given a set of *unlabeled* instances and is asked form to "useful" concept descriptions. A common approach to this problem, called *conceptual clustering* (Michalski & Stepp, 1983), involves (a) determining how to cluster the instances and (b) building descriptions for those clusters. Typically, conceptual clustering methods form a hierarchy or taxonomy of concepts. Although they are superficially similar to decision trees, each node in a concept hierarchy has an associated concept description that is used during classification. Also, although both paradigms

typically involve a top-down search from simple to more complex representations, conceptual clustering methods use different search operators and evaluations functions than those used for building decision trees. For example, some techniques measure the simplicity of the potential clusters' descriptions; others aim to maximize predictive accuracy over all attributes, since no class information is available.

Conceptual clustering is an active area of research within the empirical learning paradigm. Recent work has focused on using goals to direct the clustering process (Stepp & Michalski, 1986), on devising incremental methods for hierarchy formation (Fisher, 1987; Lebowitz, 1987), and on the role of probabilistic descriptions in clustering (Cheeseman, Kelly, Self, Stutz, Taylor, & Freeman, 1988; Fisher, 1987). These methods hold considerable promise because they address the issue of organizing concepts into integrated memory structures.

Methods for conceptual clustering appear to be appropriate for classification domains where no expert is readily available. For example, musicologists viewed the taxonomy formed by one conceptual clustering algorithm as a significant, scientific contribution (Michalski & Stepp, 1983). Similarly, a learned set of classes of infrared stellar bodies were judged significant by astronomers (Cheeseman et al., 1988). Finally, conceptual clustering methods have led to improved performance on prediction tasks for other domains, including soybean diagnosis and congressional voting records (Fisher, 1987).

3.3.4 Open Issues in Empirical Learning

Although our understanding of empirical learning methods has reached the stage where initial applications are feasible, basic research is still in progress. Active areas of research include:

- *Incremental learning.* Many existing empirical methods are nonincremental: they must reprocess many instances to incorporate new information. Incremental methods can be more efficient, but most are not yet as robust as their nonincremental counterparts (Fisher, 1989; Utgoff, 1989).
- *Search-limited methods.* Many existing empirical techniques carry out a significant search through their space of descriptions. Search-limited methods, such as hill climbing and greedy algorithms, are much more efficient, but they should be modified to increase their chances of finding satisfactory solutions (Quinlan, 1986a; Langley, Gennari, & Iba, 1987).
- *Incorporating domain knowledge.* Most empirical learning systems use domain knowledge in minimal ways. These methods should be extended to use available knowledge to constrain search and produce clearer knowledge structures (Drastal, Raatz, & Czako, 1989; Elio & Watanabe, in press; Hirsh, 1989).
- *Representation change.* Most learning systems are unable to extend their initial representation language. Empirical methods for defining new terms hold great promise, but they must constrain their search for such terms and generate effective candidates (Muggleton & Buntine, 1988; Pagallo, 1989; Matheus & Rendell, 1989).

- *Noise and concept drift.* Some methods are robust with respect to noise and changing environments (Schlimmer & Granger, 1986), but researchers should explore the general principles underlying these issues and identify the class of techniques that can handle them.

- *Uncertainty and probability.* Many expert systems for classification employ probability or other techniques for handling uncertainty, but few learning methods incorporate these ideas. Researchers need to extend existing representations and methods so as to handle uncertainty (Fisher, 1987; Geiger, Paz, & Pearl, 1990).

Research on empirical techniques is leading to continual progress on the above problems, but much remains to be done in this promising area of machine learning.

3.4 Analytic Learning

In contrast to the inductive learning methods discussed so far, another major paradigm in machine learning focuses on *analytic* learning. This approach emphasizes the transformation of existing domain knowledge into a more useful form, using data only to guide the application of deductive processes to this knowledge. These methods are sometimes called *symbol-level learners* (Dietterich, 1987) because the learned knowledge increases efficiency.[1] If the performance system must operate under limited resources, such as time or memory, then such learning can indirectly lead to improvements in accuracy as well.

In terms of the dimensions of Section 2, learning methods in this paradigm typically encode experiences, domain knowledge, and learned knowledge with relational representations. Because efficiency is the most obvious benefit of analytic learning, these methods have typically been applied to problem-solving performance tasks. They have also been studied in both supervised and unsupervised learning situations; in addition to constructing new operators, these methods have also been applied to the task of improving operator selection. As many have observed, methods of this type could be applied to existing knowledge without the aid of data to guide learning. This nonincremental approach has been predominantly shunned in favor of more computationally efficient, incremental approaches, though there are recent exceptions (Etzioni, 1990).

Explanation-based learning is one common approach that can be viewed as compiling knowledge into an efficient form rather than creating or extending knowledge. This class of learning methods can simplify a problem solver's reasoning process by composing rules into useful combinations. Explanation-based learning can also acquire control knowledge that limits alternatives, thus reducing the amount of search problem solving incurs. There currently exist number of well-specified algorithms for explanation-based learning, and recently there have been some successful applications to significant problem-solving tasks. In this section, we review the basic approach, its adaptation to problem solving, and open research questions.

1. However, in some applications, inductive methods can also improve efficiency (Langley, 1985; Mitchell et al., 1983; Ohlsson, 1987).

3.4.1 COMPILING EXPLANATIONS INTO RULES

Much of the AI research on reasoning and theorem proving takes a *problem reduction* approach. This framework assumes that domain knowledge is specified as a set of inference rules or goal decompositions. For example, to get from Washington to New York, an agent can drive to National airport, fly to LaGuardia, and take a taxi into the city. Given a top-level goal, an agent can use "and-or" search to find some set of primitive actions, states, or beliefs that will achieve that goal. Some programming languages, such as PROLOG, support this form of reasoning directly.

The result of a problem-reduction search is a "proof tree" or "explanation" for how to achieve the initial goal. Methods for explanation-based learning use this information during the learning process to create summarizations of search that can simplify future search for similar goals (DeJong & Mooney, 1986; Mitchell, Keller, & Kedar-Cabelli, 1986). More precisely, given a problem and its solution, explanation-based learning uses a proof tree for the solution of a problem to: (a) focus attention on relevant problem features, and (b) summarize the problem-solution pair as a general rule. The resulting rule states the conditions under which the proof will hold, and in the future, similar problems can be solved in fewer search steps.

Figure 5 summarizes the basic ideas that underlie explanation-based learning. This approach provides an important demonstration of the use of knowledge in learning, and the basic method can be applied to any domain in which knowledge can be stated as monotonic production rules (i.e., rules that only add knowledge). This includes problem-solving domains for which useful goal decompositions are already known, as well as many reasoning tasks and design problems.

3.4.2 LEARNING MACRO-OPERATORS

Not all research on problem solving views this process in terms of and-or search. In contrast, some work focuses on *state-space* search, in which one applies a sequence of operators to problem states in order to achieve some desired state or goal. For example, in the blocks-world domain, states and goals involve specific configurations of objects, whereas operators specify the preconditions and results of actions that manipulate objects. Explanation-based methods can use knowledge of such legal operators to construct general rules that increase problem-solving efficiency (Fikes, Hart, & Nilsson, 1972).

The most basic application of explanation-based learning to state-space search is straightforward. Once a problem-solving system has found a sequence of operators that transform an initial state to a goal, one composes this solution path into a single rule or *macro-operator* (Fikes et al., 1972; Iba, 1989). The conditions of this rule include all aspects of the initial problem state that were required for the solution to hold, and the results include all those actions not undone by others along the way. This composition process is more complex than the one described in Section 3.4.1 because operators can be nonmonotonic (i.e., they add *and* delete facts). However, it is simpler in other ways because solution paths are sequential rather than tree structured.

The construction of macro-operators lets a problem solver take larger steps through a problem space and thus shorten the effective length of solution paths. In contrast to inductive learning methods, macro construction is a purely deductive process — the form of the new rule is completely

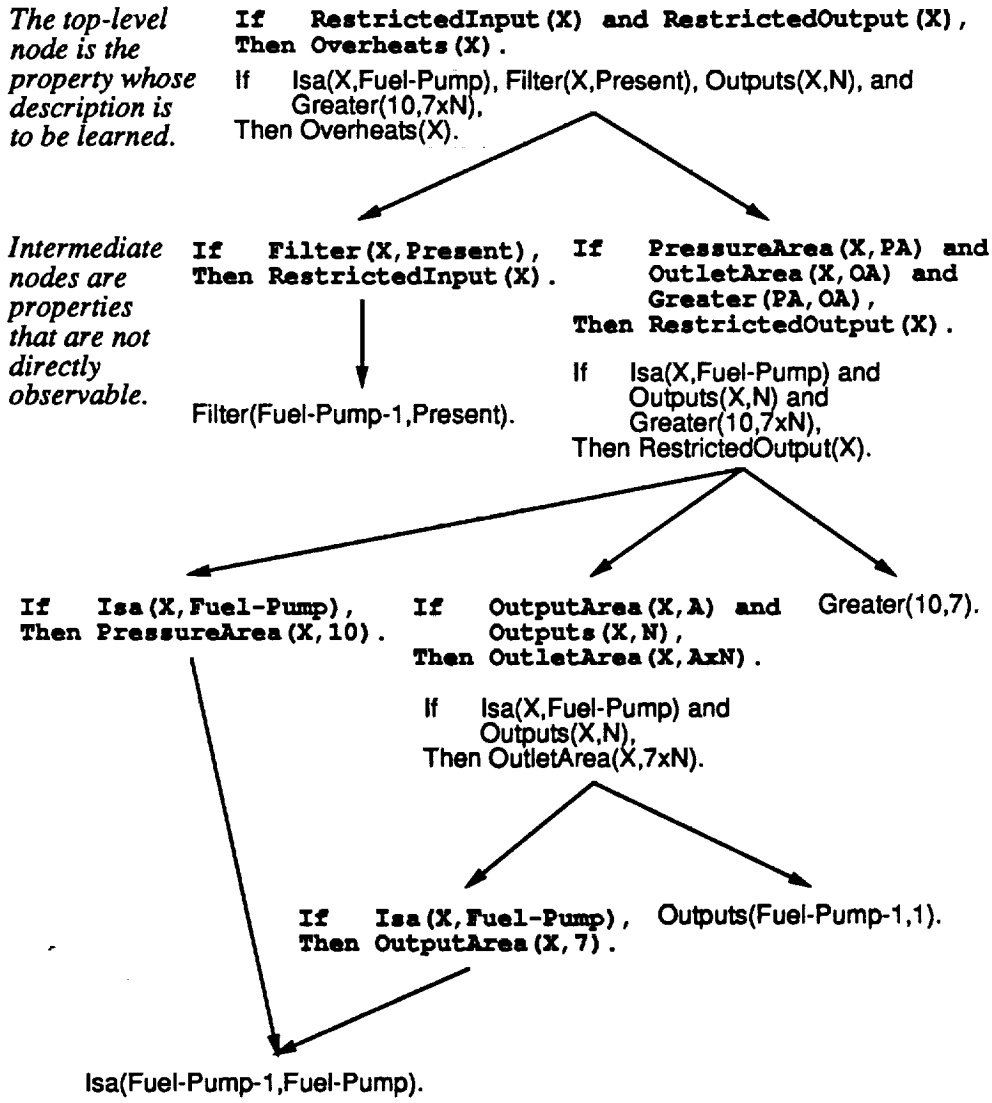

Figure 5. Schematic of the representation used by explanation-based generalization. Properties of an instance (depicted as leaves in the figure) and domain knowledge encoded as rules (depicted in teletype font) justify or *explain* other properties of the instance that may not be immediately observable. By extracting the weakest preconditions under which this justification holds, one can construct rules that are more general than the specific instance but more special than the general domain knowledge rules (depicted below each domain knowledge rule in sans-serif font). These rules allow inferring a non-observable property by testing only immediately observable properties of an instance.

determined by the primitive operators and their instantiations. Most work on discovering macro-operators has focused on abstract tasks like the eight puzzle and the blocks world, but it should apply to any task that can be cast in terms of state-space search. The same basic method also applies to planning approaches, in which subgoals are created during the problem-solving process (Minton, 1985).

3.4.3 LEARNING PREFERENCE RULES FOR PROBLEM SOLVING

Another approach to explanation-based learning generates heuristics for problem solving. Although early work on rule-based problem solving used domain-independent heuristics to select which state to expand and which operator to apply during search, some recent learning research has focused on the acquisition of domain-specific search control rules that conditionally prefer one state or operator to another. In Section 3.3.1 we saw that one could use empirical methods to learn such heuristics, but this task also fits the requirements of explanation-based methods,[2] and more work on learning search control has occurred within this paradigm.

In one version of this approach, a mean-ends planning system uses control rules to select a state to expand, an operator to apply, and a binding for the variables in the operator (Minton, Carbonell, Knoblock, Kuokka, Etzioni, & Gil, 1989). When no control rules are available, the problem solver defaults to performing a depth-first search. When search leads to failure or success, the system tries to explain that failure or success using general knowledge of problem solving. In both cases, explanation-based learning compiles the and-or explanation into a control rule that the system stores away for future use. This approach has been tested on a number of planning domains, including moderately complex scheduling tasks.

In a related but somewhat different approach, new control rules are learned only when existing control rules result in an ambiguous decision or *impasse* (Laird, Rosenbloom, & Newell, 1986). In such cases, the problem-solving system searches to determine the correct answer and compiles the result into a new control rule. Instead of using explicit knowledge about the operators themselves, the conditions of the new control rule incorporate those facts that were used in determining the correct decision, while its actions contain the results of the search. This approach has been tested on a wide range of domains, including design, puzzle solving, and a computer configuration task (Rosenbloom, Laird, McDermott, Newell, & Orciuch, 1985).

3.4.4 OPEN ISSUES IN ANALYTIC LEARNING

Although there have been rapid advances in our understanding of analytic approaches to learning, there remain many significant research issues. The current open problems include:

- *Incorrect and incomplete knowledge.* Most existing analytic methods rely on a complete and correct knowledge, but this is seldom a realistic assumption. Researchers should develop methods that behave robustly when some knowledge is missing or faulty (Laird, 1988; Smith, Winston, Mitchell, & Buchanan, 1985).
- *Extending and revising knowledge.* Another response to incomplete and incorrect domain knowledge is to extend and revise the knowledge base (Carbonell & Gil, 1987; Muggleton & Buntine, 1988; Ourston & Mooney, 1990). We need methods to detect such problems and alter the knowledge base; note that this requires some form of inductive learning, though deduction may also play a role.

2. In particular, preference rules are monotonic (i.e., they only add facts), so they may be acquired using explanation-based methods.

- *Intractable knowledge.* For useful learning to occur in this framework, the system must already be able to tractably perform search, at least until sufficient knowledge is acquired to control this search. Knowledge bases for some domains like chess are complete but intractable. What types of methods can handle approximate knowledge bases (Tadepalli, 1989), and how can this knowledge be used effectively? Some promising approaches include the use of abstraction (Ellman, 1988; Knoblock, 1990).
- *Nonlogical knowledge.* Not all domain knowledge is logical in content; in some domains one must rely on heuristic rules or probabilistic relations. Researchers should extend explanation-based methods to use such knowledge to construct explanations and to generate useful rules.
- *Evaluating alternative explanations.* Given multiple proofs, a learning method must decide which to compile into the knowledge base for future use. Researchers need to identify the basis for judging an explanation's quality, and they need to devise efficient implementation techniques for this (Ng & Mooney, 1990).
- *The utility problem.* Having selected an explanation, a learning method must still determine whether the resulting compiled rule is worth retaining. Additional knowledge can increase retrieval costs and branching factors on future problems, and learning methods must decide if the cost outweighs the benefit. Recent advances have focused on on simplifying compiled rules (Keller, 1987; Minton, 1990), limiting the expressiveness of acquired knowledge (Tambe & Rosenbloom, 1989), and collecting statistics on rule utility (Markovitch & Scott, 1989; Minton, 1990).

As with the other approaches to learning that we have examined, research on explanation-based methods continues to make steady progress, but much more remains to be done.

3.5 Case-based Methods and Analogy

There is mounting evidence that human experts rely at least partly on memory for individual cases, particularly in domains such as law, mathematics, design, and planning. Thus, it seems natural to exploit this idea in constructing AI systems, using memory of specific cases to classify new cases and to formulate plans. This is commonly called the *case-based* approach, and it constitutes a fifth major paradigm of machine learning research. Work on reasoning by analogy also falls within this general framework.

Unlike many other machine learning methods, a general theme of case-based methods is that abstraction of prior experience primarily occurs in a lazy fashion. Rather than aggressively abstracting or compiling experience in anticipation of future use, case-based methods typically save the bulk of their processing until an actual use occurs for this experience. With this deferred processing scheme, three fundamental issues arise: (a) retrieving case(s) that may help with a new case, (b) matching and applying the retrieved case(s) to the new case, and (c) storing the outcome of the new case for future use. Rather than discuss each of these issues separately, in this section we review three subparadigms within the case-based paradigm and discuss the issues as they arise.

Now let us consider the placement of case-based methods on the dimensions of Section 2. This framework represents individual experience either propositionally or with relational languages, al-

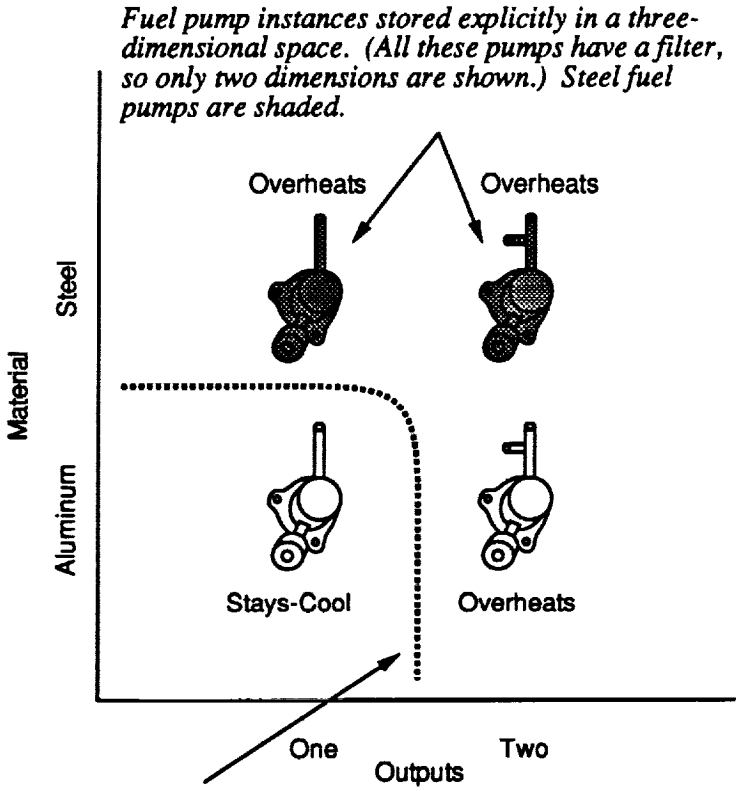

Figure 6. Schematic of the representation used in a nearest neighbor learning method for the fuel-pump domain. Instances are stored explicitly in a space of N dimensions, where each dimension describes one feature of the instance. The figure depicts the two-dimensional case. To classify an instance, one finds the nearest stored instance and predicts its class. It is possible to define a decision boundary representing equal distance from stored instances of different classes (shown as a curved line in the middle of the figure). Typically these decision boundaries are not stored explicitly.

though the latter are not as widespread because of the undesirable matching complexity that may result. Like empirical learning methods, case-based methods have been applied to both classification and problem-solving tasks; they have been applied in both supervised and unsupervised learning situations; and they are predominantly incremental. Current case-based methods always perform induction (usually at retrieval time), but many include an analytic component as well.

3.5.1 NEAREST-NEIGHBOR TECHNIQUES

In the simplest variant on the case-based framework, one simply stores past instances verbatim. When a new case is encountered, one finds the best match from among the stored instances, and then uses that case to directly supply the missing information. For example, in a medical diagnosis domain, each case would consist of a patient's symptoms along with his disease. Given a new

patient's symptoms, this simple variant enumerates stored cases to find the 'nearest neighbor' and uses its associated disease to predict the patient's malady. Figure 6 illustrates the basic technique, which originated in work on pattern recognition (Cover & Hart, 1967). Nearest neighbor algorithms are not limited to classification domains, as shown in recent work using case-based techniques for state-space problem solving (Bradtke & Lehnert, 1988).

Variants on this method are possible, such as expanding retrieval to making predictions based on a weighted average using the K nearest neighbors. Applying this technique to the task of mapping letters to phonemes achieves an 88% predictive accuracy on a test set of 1024 instances (Stanfill, 1987). The same research also demonstrates that this method degrades gracefully as one adds noise and as one decreases the size of the case base.

Uncontrolled growth of the case base is a natural concern in this paradigm. Thus, one may want to store cases selectively and delete others on occasion. In one instance of this approach, new cases are stored only when the existing knowledge base leads to a classification error (Aha, Kibler, & Albert, 1991). Although relatively unsophisticated, the predictive accuracy of this approach compares favorably to methods for inducing decision trees, and a similar approach has been successfully applied to the challenging domain of speech recognition (Bradshaw, 1987).

3.5.2 ANALOGICAL MATCHING

Given feature-based or attribute-based representations, the process of matching two cases is simple and inexpensive. However, domains like planning and design require structural or relational representations, and these introduce serious complexities into the match process. Since the new case and the stored case are unlikely to match exactly, one must perform some form of partial matching, and this problem is exponential (Watanabe & Rendell, 1990).

Researchers concerned with the process of *analogy* have devoted considerable attention to this issue, with most approaches involving some form of heuristic search through the space of partial matches. In this framework, the main issue becomes findings ways to constrain and direct the search for a useful match. For instance, one approach finds mappings that preserve higher-order relations between two cases in preference to ones that preserve simple features shared by the cases (Falkenhainer, Forbus, & Gentner, 1989). Other researchers (e.g., Winston, 1984) have proposed different but related methods. In a complementary vein, knowledge about the domain can help evaluate potential matches and identify meaningful partial matches (Koton, 1988b).

3.5.3 INDEXING AND MEMORY ORGANIZATION

Even with selective storage of cases, complex real-world domains might require thousands of instances, some having considerable structure that requires complex analogical matching. In such situations, one cannot afford to exhaustively match against all cases stored in memory, and the initial storage and subsequent retrieval of relevant cases become central issues. The natural response is to *index* cases by appropriate features, thus making the retrieval process more selective and reducing the effect of memory size. Early work on analogy focused on matching to the exclusion of indexing issues, but this has changed in recent years.

The first step in indexing cases involves selecting an appropriate set of indices. The programmer can fix the indices at the outset, but this produces an inflexible system that cannot adapt to new domains. Some researchers (e.g., Lebowitz, 1987) have invoked inductive learning methods to identify predictive features, which are then used as indices. Others have used explanation-based techniques to determine relevant features for each case and index on these instead. Problem-solving domains are especially well-suited to the latter approach, since the trace of problem-solving behavior (i.e., goal trees) provides ready-made information for explaining success or failure (Carbonell, 1986; Hammond, 1986), which can then be used to index complex cases. The notion of "derivational replay" (Mostow, 1989) based on such problem-solving traces has received attention in many circles, including software engineering and automated VLSI design.

However, indexing by itself is not sufficient to allow efficient retrieval of relevant cases. For large knowledge bases, one must also *organize* memory into some manageable structure. Discrimination networks (Feigenbaum, 1963) are one approach to memory organization, but retrieval of a case often depends on a conjunction of features being present. This leads to fragility in domains where features can be missing, but the basic approach can be extended to support redundant indexing (Kolodner, 1983; Lebowitz, 1987; Fisher, 1987). In addition, one can store abstract summary descriptions at internal nodes in the network, giving generalization beyond individual cases. The construction of 'prototypes' in this manner reveals an underlying similarity between case-based learning and conceptual clustering, which we discussed in Section 3.3.3.

3.5.4 Open Issues in Case-based Learning

Research on case-based approaches has led to a number of promising methods, some of which have been tested on challenging domains. However, a number of open issues remain to be addressed:

- *Selecting indices.* A number of methods exist for selecting indices. However, we need to identify common processes that underlie these approaches and the common information they exploit (Bareiss, Porter, & Weir, 1987; Hammond, 1986). We also need to better understand methods that generate indices dynamically (Barletta & Mark, 1988; Kolodner, 1989; Owens, 1989).
- *Memory organization.* Some initial work has addressed the organization of memory, but we need to identify the general properties that a memory should exhibit, and to improve methods for dynamically reorganizing memory as new cases are encountered (Kolodner, 1983).
- *Matching metrics.* Many existing techniques employ *ad hoc* schemes for matching against cases in memory. Researchers need to search for underlying principles involved in determining a good match and to develop methods for predicting good matches (Kolodner, 1989; Koton, 1988a; Salzberg, 1990).
- *Multiple cases.* Analogy has predominantly focused on using sophisticated information from a single case, whereas K nearest-neighbor methods illustrate how to use simple information from multiple cases. Some early work has focused on combining highly relevant but contrasting cases (Ashley & Rissland, 1988), but we need to identify other types of information that multiple cases can provide (Aha & Kibler, 1990; Redmond, 1990).

- *Connections among cases.* In some domains, each case may have a complex, internal structure, effectively consisting of many component cases. Researchers should devise principled representational schemes that can capture connections between component cases and find methods for efficiently storing and retrieving such structures (Jones, 1989; Redmond, 1990; Sycara, 1988).
- *Forgetting.* Although most cases are useful, storage of all cases can lead to overfitting effects in noisy domains. We need techniques that can efficiently determine when to forget cases (Aha, Kibler, & Albert, 1991). An alternative is to develop methods that selectively acquire knowledge, thereby limiting the discovery of useless cases and unnecessary processing (Hunter, 1990).

Work on case-based reasoning has produced some promising techniques, but researchers need to more fully explore the space of such methods, and to carefully evaluate alternative approaches in terms of their performance on real-world domains.

3.6 Relationships among the Paradigms

Historically, machine learning researchers have emphasized differences among the five paradigms we have discussed, rather than their similarities. This trend has been encouraged by differences in terminology, notation, test cases, and methods of evaluation. For instance, researchers studying connectionist techniques, genetic algorithms, and rule induction often run experiments with their inductive methods, but they typically use different data sets and measure different aspects of learning behavior. The same problem occurs between workers in the explanation-based and case-based paradigms. Our discussion so far has reflected this trend, focusing on the differences between the five learning frameworks.

However, understanding the similarities among these paradigms is equally important to the science of machine learning. To this end, let us briefly consider some possible connections:

- *Symbolic and subsymbolic induction.* Many distinguish the "subsymbolic" approach of connectionist and genetic algorithms from the "symbolic" approach taken by empirical methods for rule induction and decision-tree construction. But despite differences in the representation of acquired knowledge, the spaces searched, and the learning operators employed, all three approaches are inductive in nature, and one can generally apply them to the same learning tasks. Recent comparative studies have clarified this fact (Mooney, Shavlik, Towell, & Gove 1989; Dietterich, Hild, & Bakiri, 1990), producing comparable results for a variety of methods.
- *Induction and explanation.* Researchers often make a dichotomy between inductive (e.g., empirical) methods and analytic (e.g., explanation-based) ones, characterizing the former as "knowledge lean" and the latter as "knowledge intensive." Yet there is nothing mutually exclusive about these approaches, and hybrid methods should prove better than either in isolation. For instance, recent work on empirical methods has shown that domain knowledge and deduction can improve learning (Drastal et al., 1989; Elio & Watanabe, in press). Similarly, one can use empirical methods to extend incomplete domain knowledge and to revise incorrect rules (Carbonell & Gil, 1987; Ourston & Mooney, 1990).

- *Explanations and cases.* We have seen that explanation-based methods use domain knowledge to construct explanations and compile rules, but some case-based techniques rely just as heavily on domain expertise (Braverman & Wilensky, 1990; Redmond, 1989). Although one approach stores general rules and the other stores specific cases, the reasoning processes can be remarkably similar.

- *Cases and abstractions.* Researchers often emphasize the distinction between storing specific cases and forming abstractions. Yet many case-based systems create abstractions as indices for cases (Kolodner, 1983; Fisher, 1987), making them more accurately described as hybrids. Nor must a system that stores cases always use them during performance; in some situations, a hybrid system may prefer to use an abstraction (Fisher, 1989). Moreover, work on conceptual clustering and on case-based learning shares a concern with the organization of memory, relying on similar structures and mechanisms.

In summary, there is considerable overlap between the paradigms in both their concerns and their approaches, although this is seldom apparent from research papers. Machine learning has just begun to converge on a set of standard terms and notations for describing systems, and on a set of standard testbeds and experimental methodologies for evaluating systems.

As researchers start to communicate across paradigm boundaries, they can begin exploring the relationships more seriously. For instance, one can imagine a unified theory of induction that explains the behavior of decision-tree methods, genetic algorithms, and connectionist networks, and that predicts the conditions under which each method would be most appropriate. One can also expect the development of hybrid algorithms that cut across paradigms to achieve better results than either in isolation. Some research along these lines has already begun, such as recent work on combining decision trees with perceptrons (Utgoff, 1988). We hope that some researchers will concentrate their efforts on such cross-paradigm research, since this may lead to new techniques that otherwise might never come to light. If such systems were successful, this would further strengthen the ties between areas, ultimately transforming machine learning into a unified field rather than one composed of many subdisciplines.

4. Methodological Developments in Machine Learning

Machine learning is a scientific discipline, and thus careful methodological foundations are essential to its success. Over the past few years, some significant methodological advances have occurred in the field, paving the way for more careful work in the future. These include new techniques for the formal analysis of learning algorithms, new experimental approaches to studying learning, successful applications to real-world domains, and the development of integrated cognitive architectures. These changes bode well for this emerging subfield of artificial intelligence, and we discuss each of them briefly below.

4.1 Theoretical Analyses of Learning Algorithms

Although formal studies of inductive inference have a long history in computer science (Angluin & Smith, 1983), only recently have theorists started to address issues of concern to researchers who actually build machine learning systems. Initial ideas focused on the notion of convergence: Would the learning method eventually construct the exact knowledge desired (Gold, 1967)? This approach constituted an important first step, but it did not afford much insight into realistic learning problems.

A major breakthrough came when researchers turned their focus to the question of evaluating the *quality* of inductively learned knowledge. The notion of *probably approximately correct* (PAC) learning forwarded the idea that learned knowledge should usually be relatively accurate when applied in novel situations. Coupling this idea with computational feasibility yields a definition for problems that are *polynomially learnable*: (a) it must not require too many instances to learn, (b) there must exist an efficient learning method which can produce PAC knowledge, and (c) it must be possible to efficiently determine whether knowledge is consistent with any given instance (Blumer, Ehrenfeucht, Haussler, & Warmuth, 1987).

When studying the learnability of a particular problem, one common research tactic is to show that one of the three criteria cannot be met. However, because each of these three criteria depends on how acquired knowledge is represented, results of negative learnability can be brittle. For instance, learning K of N functions using a linear threshold unit with only zero or one weights is not polynomially learnable, but the same class of functions *are* learnable if the linear threshold unit can use integer weights (Haussler, 1990).

The initial theoretical frameworks focused on learning logical, feature-based concepts in a supervised setting (Kearns, Li, Pitt, & Valiant, 1987), but researchers have since extended the basic framework to other paradigms, including structural concepts (Haussler, 1987), decision lists (Rivest, 1987), conceptual clustering (Pitt & Reinke, 1988), and connectionist networks (Valiant, 1988). They have also addressed learning in the presence of noise (Angluin & Laird, 1988), and they have moved beyond inductive methods to deal with explanation-based methods (Natarajan & Tadepalli, 1988). Even many nontheorists follow this work closely, and many theorists actively read the empirical literature in search of challenging problems.

4.2 Experimental Studies of Learning Algorithms

Despite progress on the theoretical front, many learning algorithms remain too complex for formal analysis, and recent progress has also been made in the experimental study of learning methods (Kibler & Langley, 1988). Much of the experimental work has focused on inductive methods (Fahlman, 1988; Fisher, 1987; Quinlan, 1986b; Schlimmer, 1987), but there are also a growing number of experimental studies of explanation-based techniques (Minton et al., 1989; Shavlik, 1990).

One important insight is that *performance* is the natural dependent measure for such empirical studies, since one can define learning as improvement in performance. There are many measures of performance, including classification accuracy, quality of solution paths, and even CPU time.

Different measures are appropriate for different domains and different learning methods, since they may have different goals. However, having at least some measure of performance is essential to evaluating a learning system's behavior. In some cases, intuitively plausible learning methods actually lead to worse performance (Minton, 1985). In other cases, one can use performance measures to determine which components significantly aid the learning process (Schlimmer, 1987).

Researchers have also started to carefully examine the aspects of domains that affect learning behavior. Some experimental studies have focused on naturalistic data in order to show real-world relevance, but others have constructed synthetic domains to allow control of domain characteristics. Two obvious features include complexity of the knowledge to be learned and amount of noise in the data, but others certainly exist. The important point is that many researchers now realize that, in order to make progress, the field requires some explicit methods for evaluating alternative methods and for identifying the conditions under which they work well. Theoretical analyses provide one route to such understanding, but systematic experimentation is another important path.

4.3 Common Testbeds and Applications

Early research in machine learning focused on idealized, hand-crafted examples, and researchers often tested their systems on only a handful of cases. This has changed drastically in recent years, and papers in the literature now commonly report results on realistic learning tasks that involve many test cases. Moreover, researchers typically report results on a number of different data sets, to show the robustness and generality of their algorithms. The average number of test domains should increase as the standards of the field become higher.

Another encouraging sign is that researchers are starting to test their algorithms on the *same* task domains, allowing comparisons to be made. This trend has been aided by the collection and distribution of standard data sets. For instance, data on soybean diseases (Michalski & Chilausky, 1980), thyroid diseases (Quinlan, 1987), edibility of mushrooms (Schlimmer, 1987), and Congressional voting records (Fisher, 1987) have been widely distributed and used in testing a number of learning algorithms. Major repositories have emerged, with researchers collecting, documenting, and distributing benchmark data sets. Many of these deal with classification and diagnosis, but standard problem-solving and reasoning tasks are also beginning to emerge.

Despite these encouraging developments, most of these real-world domains remain relatively simple and straightforward. We hope that future application efforts will tackle more difficult testbeds that provide greater challenges for machine learning methods. We also hope the trend will expand to include the documentation and distribution of published algorithms, so that researchers can employ each others' software. This has started to occur within some machine learning paradigms, but more remains to be done.

4.4 Integrated Cognitive Architectures

Another methodological advance relates to the development of integrated architectures for cognition. Early AI researchers commonly implemented a separate system for each new task they encountered. As the field gained experience, high-level languages (e.g., production systems) were

developed and used to implement new systems, with considerable savings in time and effort. However, these languages incorporated only minor theoretical commitments about the nature of intelligent behavior, and thus provided few constraints on the resulting AI systems. For instance, few formalisms included any automated learning mechanisms.

This trend has changed in recent years, with many researchers now turning to integrated architectures that make strong assumptions about the control structures needed to support intelligence. SOAR is a prime example of this approach (Laird et al., 1986), and the classifier systems of the genetic algorithm community constitute another instance. Most work in this growing movement includes some automated learning mechanism as an integral part of the architecture, and generality is a central concern, with researchers testing their frameworks on a variety of domains. For instance, the PRODIGY (Minton et al., 1989) and THEO (Mitchell, Allen, Chalasani, Cheng, Etzioni, Ringuette, & Schlimmer, in press) architectures incorporate explanation-based methods into their problem-solving engines, whereas ICARUS (Langley, Thompson, Iba, Gennari, & Allen, in press) relies on case-based concept formation as its main learning mechanism, and DYNA (Sutton, 1990) uses connectionist learning methods. Such integrated frameworks will be necessary if we ever hope to construct intelligent artifacts that can interact with the physical world, and we predict that learning will occupy a central role in successful cognitive architectures.

5. Summary

Over the last decade, the theoretical and methodological advances described in the previous sections have transported machine learning from the sidelines of AI into one of its central foci. Along with this shift has come increased contact with other subcommunities, and as methods for machine learning become more robust, they are gaining increased attention from researchers concerned with planning, diagnosis, natural language, and other problem-oriented areas of artificial intelligence. In turn, these domains provide significant real-world challenges for scientists who have traditionally been concerned with abstract issues in machine learning.

Without doubt, the growing concern with applications will reveal limitations of the existing paradigms and suggest novel directions for automating the acquisition of knowledge. Thus, researchers will be forced to devise new representations, search frameworks, and control schemes to support the learning process. The resulting approaches may initially be domain specific, inefficient, and inelegant, but they will respond to issues that have been previously ignored. Such learning methods may not fit nicely into the organization we have presented, but that is often the nature of scientific progress.

At the same time, others will continue to pursue basic research on learning mechanisms, driven by recognized open issues like those we listed above. These scientists will explore variations and hybrids of existing methods, propose frameworks that unify apparently different techniques, and carry out experimental and theoretical studies to identify the behavior of alternative methods under varying conditions. They should also begin to relate experimental results to those predicted by theory, revising the theory when necessary. Finally, they will attempt to identify new dimensions

and new themes that have emerged from the applied work, idealizing them in ways that lets them be studied in the same manner as existing paradigms.

Taken together, basic and applied research in this area should continue to improve the range and capabilities of learning algorithms, and to increase our understanding of mechanisms for improving performance with experience. These advances in turn will have far-ranging implications for the rest of artificial intelligence, letting the field move beyond static systems to ones that change their behavior over time as they acquire and refine knowledge.

Acknowledgements

We appreciate the thoughtful comments and suggestions of Andrew Barto, Gerald DeJong, John Grefenstette, Janet Kolodner, Edwina Rissland, and Robert Simpson. Any errors or ambiguities are the sole responsibility of the authors.

References

Ackley, D. H., Hinton, G. E., & Sejnowski, T. J. (1987). A learning algorithm for Boltzmann machines. *Cognitive Science*, *9*, 147–169.

Aha, D. W., & Kibler, D. (1990). Noise-tolerant instance-based learning algorithms. *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence* (pp. 794–799). Detroit, MI: Morgan Kaufmann.

Aha, D. W., Kibler, D., & Albert, M. K. (1991). Instance-based learning algorithms. *Machine Learning*, *6*, 37–66.

Angluin, D., & Laird, P. (1988). Learning from noisy examples. *Machine Learning*, *2*, 343–370.

Angluin, D., & Smith C. (1983). Inductive inference: Theory and methods. *Computing Surveys*, *15*, 237–269.

Ashley, K. D., & Rissland, E. L. (1988). Waiting on weighting: A symbolic least commitment approach. *Proceedings of the Seventh National Conference on Artificial Intelligence* (pp. 239–244). St. Paul, MN: AAAI Press.

Bareiss, E. R., Porter, B. W., & Wier, C. C. (1987). PROTOS: An exemplar-based learning apprentice. *Proceedings of the Fourth International Workshop on Machine Learning* (pp. 12–23). Irvine, CA: Morgan Kaufmann.

Barletta, R., & Mark, W. (1988). Explanation-based indexing of cases. *Proceedings of the Seventh National Conference on Artificial Intelligence* (pp. 541–546). St. Paul, MN: AAAI Press.

Barto, A. G., Sutton, R. S., & Anderson, C. W. (1983). Neuronlike elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, *13*, 835–846.

Belew, R. K., & Forrest, S. (1988). Learning and programming in classifier systems. *Machine Learning*, *3*, 193–223.

Blumer, A., Ehrenfeucht, A., Haussler, D., & Warmuth, M. K. (1987). Occam's razor. *Information Processing Letters*, *24*, 377–380.

Booker, L. B. (1988). Classifier systems that learn internal world models. *Machine Learning, 3*, 161–192.

Bradshaw, G. (1987). Learning about speech sounds: The NEXUS project. *Proceedings of the Fourth International Workshop on Machine Learning* (pp. 1–11). Irvine, CA: Morgan Kaufmann.

Bradtke, S., & Lehnert, W. G. (1988). Some experiments with case-based search. *Proceedings of the Seventh National Conference on Artificial Intelligence* (pp. 133–138). St. Paul, MN: AAAI Press.

Braverman, M. S., & Wilensky, R. (1990). Toward a unification of case-based reasoning and explanation-based learning. *Proceedings of the 1990 AAAI Spring Symposium on Case-Based Reasoning.* Stanford, CA.

Brieman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and regression trees.* Belmont: Wadsworth.

Carbonell, J. G. (1986). Derivational analogy: A theory of reconstructive problem solving and expertise acquisition. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach* (Vol. 2). San Mateo: Morgan Kaufmann.

Carbonell, J. G., & Gil, Y. (1987). Learning by experimentation. *Proceedings of the Fourth International Workshop on Machine Learning* (pp. 256–266). Irvine, CA: Morgan Kaufmann.

Cheeseman, P., Kelly, J., Self, M., Stutz, J., Taylor, W., & Freeman, D. (1988). AUTOCLASS: A Bayesian classification system. *Proceedings of the Fifth International Conference on Machine Learning* (pp. 54–64). Ann Arbor, MI: Morgan Kaufman.

Clark, P., Niblett, T. (1989). The CN2 induction algorithm. *Machine Learning, 3*, 261–284.

Cover, T. R., & Hart, P. E. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory, 13*, 21–27.

Davis, R., & Lenat, D. B. (1982). *Knowledge-based systems in artificial intelligence.* New York: McGraw-Hill.

DeJong, G., & Mooney, R. J. (1986). Explanation-based learning: An alternative view. *Machine Learning, 1*, 145–176.

Dietterich, T. G. (1987). Learning at the knowledge level. *Machine Learning, 1*, 287–316.

Dietterich, T. G., Hild, H., & Bakiri, G. (1990). A comparative study of ID3 and backpropagation for English text-to-speech mapping. *Proceedings of the Seventh International Conference on Machine Learning* (pp. 24–31). Austin, TX: Morgan Kaufmann.

Drastal, G., Czako, G., & Raatz, S. (1989). Induction in an abstraction space: A form of constructive induction. *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence* (pp. 708–712). Detroit, MI: Morgan Kaufmann.

Elio, R., & Watanabe, L. (in press). An incremental deductive strategy for controlling constructive induction in learning from examples. *Machine Learning.*

Ellman, T. (1988). Approximate theory formation: An explanation-based approach. *Proceedings of the Seventh National Conference on Artificial Intelligence* (pp. 570–574). St. Paul, MN: AAAI Press.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science, 14*, 179–211.

Etzioni, O. (1990). Why PRODIGY/EBL works. *Proceedings of the Eighth National Conference on Artificial Intelligence* (pp. 916–922). Boston, MA: AAAI Press.

Fahlman, S. E. (1988). Faster-learning variations on back-propagation: An empirical study. *Proceedings of the 1988 Connectionist Models Summer School* (pp. 38–51). Pittsburgh, PA: Morgan Kaufmann.

Fahlman, S. E., & Lebiere, C. (1990). *The cascade-correlation learning architecture* (Tech. Rep. No. CMU-CS-90-100). Pittsburgh, PA: Carnegie Mellon University, School of Computer Science.

Falkenhainer, B., Forbus, K. D., & Gentner, D. (1989). The structure-mapping engine: Algorithm and examples. *Artificial Intelligence, 41*, 1–63.

Feigenbaum, E. A. (1963). The simulation of verbal learning behavior. In E. A. Feigenbaum & J. Feldman (Eds.), *Computers and thought.* New York: McGraw-Hill.

Fikes, R. E., Hart, P. E., & Nilsson, N. J. (1972). Learning and executing generalized robot plans. *Artificial Intelligence, 3*, 251–288.

Fisher, D. H. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine Learning, 2*, 139–172.

Fisher, D. H. (1989). Noise-tolerant conceptual clustering. *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence* (pp. 825–830). Detroit, MI: Morgan Kaufmann.

Fisher, D. H., & Langley, P. (1985). Approaches to conceptual clustering. *Proceedings of the Ninth International Joint Conference on Artificial Intelligence* (pp. 691–697). Los Angeles, CA: Morgan Kaufmann.

Fisher, D., & McKusick, K. B. (1989). An empirical comparison of ID3 and back-propagation. *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence* (pp. 788–793). Detroit, MI: Morgan Kaufmann.

Geiger, D., Paz, A., & Pearl, J. (1990). Learning causal trees from dependency information. *Proceedings of the Eighth National Conference on Artificial Intelligence* (pp. 770–776). Boston, MA: AAAI Press.

Gold, E. (1967). Language identification in the limit. *Information and Control, 16*, 447–474.

Goldberg, D. E. (1985). Genetic algorithms and rule learning in dynamic system control. *Proceedings of the First International Conference on Genetic Algorithms and Their Applications* (pp. 8–15). Pittsburgh, PA: Lawrence Erlbaum.

Goldberg, D. E. (1990). *Genetic algorithms in search, optimization, and machine learning.* Reading, MA: Addison-Wesley.

Gordon, D. F., & Grefenstette, J. J. (1990). Explanations of empirically derived reactive plans. *Proceedings of the Seventh International Conference on Machine Learning* (pp. 198–203). Austin, TX: Morgan Kaufmann.

Grefenstette, J. J. (1988). Credit assignment in rule discovery systems based on genetic algorithms. *Machine Learning*, *3*, 225–245.

Grefenstette, J. J., Ramsey, C. L., & Schultz, A. C. (1990). Learning sequential decision rules using simulation models and competition. *Machine Learning*, *5*, 355–381.

Hammond, K. (1986). Learning to anticipate and avoid planning problems through the explanation of failures. *Proceedings of the Fifth National Conference on Artificial Intelligence* (pp. 556–560). Philadelphia, PA: AAAI Press.

Hampson, S. E., & Volper, D. J. (1987). Disjunctive models of Boolean category learning. *Biological Cybernetics*, *56*, 121–137.

Haussler, D. (1987). Bias, version spaces, and Valiant's learning framework. *Proceedings of the Fourth International Workshop on Machine Learning* (pp. 324–336). Irvine, CA: Morgan Kaufmann.

Haussler, D. (1990). Probably approximately correct learning. *Proceedings of the Eighth National Conference on Artificial Intelligence* (pp. 1101–1108). Boston, MA: AAAI Press.

Hinton, G. E. (1986). Learning distributed representations of concepts. *Proceedings of the Eighth Annual Conference of the Cognitive Science Society* (pp. 1–12). Amherst, MA: Lawrence Erlbaum.

Hinton, G. E. (1989). Connectionist learning procedures. *Artificial Intelligence*, *40*, 185–234.

Hirsh, H. (1989). Combining empirical and analytical learning with version spaces. *Proceedings of the Sixth International Workshop on Machine Learning* (pp. 29–33). Ithaca, NY: Morgan Kaufmann.

Holland, J. H. (1985). Properties of the bucket brigade algorithm. *Proceedings of the First International Conference on Genetic Algorithms and Their Applications* (pp. 1–7). Pittsburgh, PA: Lawrence Erlbaum.

Hunter, L. (1990). Planning to learn. *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society* (pp. 261–268). Cambridge, MA: Lawrence Erlbaum.

Iba, G. A. (1989). A heuristic approach to the discovery of macro-operators. *Machine Learning*, *3*, 285–317.

Jones, R. (1989). *A model of retrieval in problem solving* (Tech. Rep. No. 89-27). Doctoral dissertation. Irvine: University of California, Department of Information and Computer Science.

Kearns, M., Li, M., Pitt, L., & Valiant, L. G. (1987). Recent results on Boolean concept learning. *Proceedings of the Fourth International Workshop on Machine Learning* (pp. 337–352). Irvine, CA: Morgan Kaufmann.

Keller, R. M. (1987). Concept learning in context. *Proceedings of the Fourth International Workshop on Machine Learning* (pp. 91–102). Irvine, CA: Morgan Kaufmann.

Kibler, D., & Langley, P. (1988). Machine learning as an experimental science. *Proceedings of the Third European Working Session on Learning* (pp. 81–92). Glasgow: Pittman.

Knight, K. (1989). *A gentle introduction to subsymbolic computation: Connectionism for the AI researcher* (Tech. Rep. No. CMU-CS-89-150). Pittsburgh, PA: Carnegie Mellon University, School of Computer Science.

Knoblock, C. (1990). Learning abstraction hierarchies for problem solving. *Proceedings of the Eighth National Conference on Artificial Intelligence* (pp. 923–928). Boston, MA: AAAI Press.

Kohonen, T., Oja, E., & Lehtiö, P. (1981). Storage and processing of information in distributed associative memory systems. In G. E. Hinton & J. A. Anderson (Eds.), *Parallel models of associative memory.* Hillsdale, NJ: Lawrence Erlbaum.

Kolodner, J. L. (1983). Maintaining organization in a dynamic long-term memory. *Cognitive Science, 7,* 243–280.

Kolodner, J. L. (1989). Selecting the best case for a case-based reasoner. *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society* (pp. 155–162). Ann Arbor, MI: Lawrence Erlbaum.

Kononenko, I., Bratko, I., & Roskar, E. (1984). *Experiments in automatic learning of medical diagnostic rules* (Tech. Rep.). Ljubljana, Yugoslavia: Josef Stefan Institute.

Koton, P. (1988a). Integrating case-based and causal reasoning. *Proceedings of the Tenth Annual Conference of the Cognitive Science Society* (pp. 167–173). Montreal, Quebec, Canada: Lawrence Erlbaum.

Koton, P. (1988b). Reasoning about evidence in causal explanation. *Proceedings of the Seventh National Conference on Artificial Intelligence* (pp. 256–261). St. Paul, MN: AAAI Press.

Koza, J. R. (1989). Hierarchical genetic algorithms operating on populations of computer programs. *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence* (pp. 768–774). Detroit, MI: Morgan Kaufmann.

Laird, J. E. (1988). Recovery from incorrect knowledge in SOAR. *Proceedings of the Seventh National Conference on Artificial Intelligence* (pp. 618–623). St. Paul, MN: AAAI Press.

Laird, J. E., Rosenbloom, P. S., & Newell, A. (1986). Chunking in SOAR: The anatomy of a general learning mechanism. *Machine Learning, 1,* 11–46.

Langley, P. (1985). Learning to search: From weak methods to domain-specific heuristics. *Cognitive Science, 9,* 217–260.

Langley, P. (1989). Unifying themes in empirical and explanation-based learning. *Proceedings of the Sixth International Workshop on Machine Learning.* Ithaca, NY: Morgan Kaufmann.

Langley, P., Gennari, J. H., & Iba, W. (1987). Hill-climbing theories of learning. *Proceedings of the Fourth International Workshop on Machine Learning* (pp. 312–323). Irvine, CA: Morgan Kaufmann.

Langley, P., Thompson, K., Iba, W., Gennari, J. H., & Allen, J. A. (in press). An integrated cognitive architecture for autonomous agents. In W. Van De Velde (Ed.), *Representation and learning in autonomous agents.* Amsterdam: North Holland.

Lebowitz, M. (1987). Experiments with incremental concept formation: UNIMEM. *Machine Learning, 2*, 103–138.

Marcus, S. (Ed.). (1988). *Automating knowledge acquisition for expert systems*. Boston: Kluwer.

Markovitch, S., & Scott, P. D. (1989). Utilization filtering: A method for reducing the inherent harmfulness of deductively learned knowledge. *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence* (pp. 738–743). Detroit, MI: Morgan Kaufmann.

Matheus, C. J., & Rendell, L. A. (1989). Constructive induction on decision trees. *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence* (pp. 645–650). Detroit, MI: Morgan Kaufmann.

Michalski, R. S. (1983). A theory and methodology of learning from examples. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach* (Vol. 1). San Mateo, CA: Morgan Kaufmann.

Michalski, R. S. (1987). How to learn imprecise concepts: A method for employing a two-tiered knowledge representation in learning. *Proceedings of the Fourth International Workshop on Machine Learning* (pp. 50–58). Irvine, CA: Morgan Kaufmann.

Michalski, R. S., & Chilausky, R. L. (1980). Learning by being told and learning from examples: An experimental comparison of two methods of knowledge acquisition in the context of developing an expert system for soybean disease diagnosis. *International Journal of Policy Analysis and Information Systems, 4*.

Michalski, R. S., & Stepp, R. (1983). Learning from observation: Conceptual clustering. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach* (Vol. 1). San Mateo, CA: Morgan Kaufmann.

Minsky, M., & Papert, S. (1969). *Perceptrons: An introduction to computational geometry*. Cambridge, MA: MIT Press.

Minton, S. N. (1985). Selectively generalizing plans for problem solving. *Proceedings of the Ninth International Joint Conference on Artificial Intelligence* (pp. 596–599). Los Angeles, CA: Morgan Kaufmann.

Minton, S. N. (1990). Quantitative results concerning the utility of explanation-based learning. *Artificial Intelligence, 42*, 363–391.

Minton, S. N., Carbonell, J. G., Knoblock, C. A., Kuokka, D. R., Etzioni, O., & Gil, Y. (1989). Explanation-based learning: A problem solving perspective. *Artificial Intelligence, 40*, 63–118.

Mitchell, T. M. (1977). Version spaces: A candidate elimination approach to rule learning. *Proceedings of the Fifth International Joint Conference on Artificial Intelligence* (pp. 305–310). Cambridge, MA: Morgan Kaufmann.

Mitchell, T. M., Allen, J., Chalasani, P., Cheng, J., Etzioni, O., Ringuette, M., & Schlimmer, J. C. (in press). THEO: A framework for self-improving systems. In K. VanLehn (Ed.), *Architectures for intelligence*. Hillsdale, NJ: Lawrence Erlbaum.

Mitchell, T. M., Keller, R. M., & Kedar-Cabelli, S. T. (1986). Explanation-based generalization: A unifying view. *Machine Learning, 1*, 47–80.

Mitchell, T. M., Mahadevan, S., & Steinberg, L. (1985). LEAP: A learning apprentice for VLSI design. *Proceedings of the Ninth International Joint Conference on Artificial Intelligence* (pp. 573–580). Los Angeles, CA: Morgan Kaufmann.

Mitchell, T. M., Utgoff, P. E., & Banerji, R. B. (1983). Learning problem solving heuristics by experimentation. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach* (Vol. 1). San Mateo, CA: Morgan Kaufmann.

Mooney, R. J., Shavlik, J. W., Towell, G. G., & Gove, A. (1990). An experimental comparison of symbolic and connectionist learning algorithms. *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence* (pp. 775–780). Detroit, MI: Morgan Kaufmann.

Mostow, J. (1989). Design by derivational analogy: Issues in the automated replay of design plans. *Artificial Intelligence, 40*, 119–184.

Mozer, M., & Bachrach, J. (in press). SLUG: A connectionist architecture for inferring the structure of finite-state environments. *Machine Learning.*

Muggleton, S., & Buntine, W. (1988). Machine invention of first-order predictates by inverting resolution. *Proceedings of the Fifth International Conference on Machine Learning* (pp. 339–352). Ann Arbor, MI: Morgan Kaufmann.

Natarajan, B. K., & Tadepalli, P. (1988). Two new frameworks for learning. *Proceedings of the Fifth International Conference on Machine Learning* (pp. 402–415). Ann Arbor, MI: Morgan Kaufmann.

Ng, H. T., & Mooney, R. J. (1990). On the role of coherence in abductive explanation. *Proceedings of the Eighth National Conference on Artificial Intelligence* (pp. 337–342). Boston, MA: AAAI Press.

Nilsson, N. (1965). *Learning machines.* New York: McGraw-Hill.

Ohlsson, S. (1987). Transfer of training in procedural learning: A matter of conjectures and refutations? In L. Bolc (Ed.), *Computational models of learning*. Berlin: Springer-Verlag.

Ourston, D., & Mooney, R. J. (1990). Changing the rules: A comprehensive approach to theory refinement. *Proceedings of the Eighth National Conference on Artificial Intelligence* (pp. 815–820). Boston, MA: AAAI Press.

Owens, C. (1989). Integrating feature extraction and memory search. *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society* (pp. 163–170). Ann Arbor, MI: Lawrence Erlbaum.

Pagallo, G. (1989). Learning DNF by decision trees. *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence* (pp. 639–644). Detroit, MI: Morgan Kaufmann.

Pitt, L., & Reinke, R. E. (1988). Criteria for polynomial-time (conceptual) clustering. *Machine Learning, 2*, 371–396.

Prager, R., Harrison, T. D., & Fallside, F. (1986). Boltzmann machines for speech recognition. *Computer Speech and Language, 1*, 1–20.

Quinlan, J. R. (1983). Learning efficient classification procedures and their application to chess endgames. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach* (Vol. 1). San Mateo, CA: Morgan Kaufmann.

Quinlan, J. R. (1986a). Induction of decision trees. *Machine Learning, 1*, 81–106.

Quinlan, J. R. (1986b). The effect of noise on concept learning. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach* (Vol. 2). San Mateo, CA: Morgan Kaufmann.

Quinlan, J. R. (1987). Generating production rules from decision trees. *Proceedings of the Tenth International Joint Conference on Artificial Intelligence* (pp. 304–307). Milan, Italy: Morgan Kaufmann.

Quinlan, J. R., Compton, P. J., Horn, K. A., & Lazarus, L. (1986). Inductive knowledge acquisition: A case study. *Proceedings of the Second Australian Conference on Applications of Expert Systems*. Sydney, Australia.

Redmond, M. (1989). Combining case-based reasoning, explanation-based learning, and learning from instruction. *Proceedings of the Sixth International Workshop on Machine Learning* (pp. 20–22). Ithaca, NY: Morgan Kaufmann.

Redmond, M. (1990). Distributed cases for case-based reasoning; facilitating use of multiple cases. *Proceedings of the Eighth National Conference on Artificial Intelligence* (pp. 304–309). Boston, MA: AAAI Press.

Rivest, R. L. (1987). Learning decision lists. *Machine Learning, 2*, 229–246.

Robertson, G. G. (1988). Population size in classifier systems. *Proceedings of the Fifth International Conference on Machine Learning* (pp. 142–152). Ann Arbor, MI: Morgan Kaufmann.

Rosenblatt, F. (1962). *Principles of neurodynamics*. New York: Spartan Books.

Rosenbloom, P. S., Laird, J. E., McDermott, J., Newell, A., & Orciuch, E. (1985). R1-SOAR: An experiment in knowledge-intensive programming in a problem-solving architecture. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 7*, 561–569.

Rumelhart, D. E., & McClelland, J. L. (1986). On learning the past tenses of English verbs. In J. L. McClelland & D. E. Rumelhart (Eds.), *Parallel distributed processing: Explorations in the microstructure of cognition* (Vol. 2). Cambridge, MA: MIT Press.

Salzberg, S. (1990). *Learning with nested generalized exemplars*. Boston, MA: Kluwer.

Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal on Research and Development, 3*, 210–229.

Schaffer, J. D., & Grefenstette, J. J. (1985). Multi-objective learning via genetic algorithms. *Proceedings of the Ninth International Joint Conference on Artificial Intelligence* (pp. 593–595). Los Angeles, CA: Morgan Kaufmann.

Schlimmer, J. C. (1987). Incremental adjustment of representations for learning. *Proceedings of the Fourth International Workshop on Machine Learning* (pp. 79–90). Irvine, CA: Morgan Kaufmann.

Schlimmer, J. C., & Fisher, D. H. (1986). A case study of incremental concept induction. *Proceedings of the Fifth National Conference on Artificial Intelligence* (pp. 496–501). Philadelphia, PA: AAAI Press.

Schlimmer, J. C., & Granger, R. H., Jr. (1986). Beyond incremental processing: Tracking concept drift. *Proceedings of the Fifth National Conference on Artificial Intelligence* (pp. 502–507). Philadelphia, PA: AAAI Press.

Sejnowski, T. J., & Rosenberg, C. R. (1987). Parallel networks that learn to pronounce English text. *Complex Systems, 1*, 145–168.

Shaefer, C. G. (1987). The ARGOT strategy: Adaptive representation genetic optimizer technique. *Proceedings of the Second International Conference on Genetic Algorithms* (p. 50).

Shavlik, J. W. (1990). Acquiring recursive and iterative concepts with explanation-based learning. *Machine Learning, 5*, 39–70.

Sleeman, D., Langley, P., & Mitchell, T. (1982). Learning from solution paths: An approach to the credit assignment problem. *AI Magazine, 3*, 48–52.

Smith, R. G., Winston, H. A., Mitchell, T. M., & Buchanan, B. G. (1985). Representation and use of explicit justifications for knowledge base refinement. *Proceedings of the Ninth International Joint Conference on Artificial Intelligence* (pp. 673–680). Los Angeles, CA: Morgan Kaufmann.

Smith, S. F. (1983). Flexible learning of problem solving heuristics through adaptive search. *Proceedings of the Eighth International Joint Conference on Artificial Intelligence* (pp. 422–425). Karlsruhe, Germany: Morgan Kaufmann.

Stanfill, C. W. (1987). Memory-based reasoning applied to English pronunciation. *Proceedings of the Sixth National Conference on Artificial Intelligence* (pp. 577–581). Seattle, WA: AAAI Press.

Stepp, R. E., & Michalski, R. S. (1986). Conceptual clustering: Inventing goal-oriented classifications of structured objects. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach* (Vol. 2). San Mateo: Morgan Kaufmann.

Sutton, R. S. (1990). Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. *Proceedings of the Seventh International Conference on Machine Learning* (pp. 216–224). Austin, TX: Morgan Kaufmann.

Sycara, K. (1988). Patching up old plans. *Proceedings of the Tenth Annual Conference of the Cognitive Science Society* (pp. 405–411). Montreal, Quebec, Canada: Lawrence Erlbaum.

Tadepalli, P. (1989). Planning in games using approximately learned macros. *Proceedings of the Sixth International Workshop on Machine Learning* (pp. 221–223). Ithaca, NY: Morgan Kaufmann.

Tambe, M., & Rosenbloom, P. (1989). Eliminating expensive chunks by restricting expressiveness. *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence* (pp. 731–737). Detroit, MI: Morgan Kaufmann.

Towell, G. G., Shavlik, J. W., & Noordewier, M. O. (1990). Refinement of approximate domain theories by knowledge-based neural networks. *Proceedings of the Eighth National Conference on Artificial Intelligence* (pp. 861–866). Boston, MA: AAAI Press.

Utgoff, P. E. (1988). Perceptron trees: A case study in hybrid concept representations. *Proceedings of the Seventh National Conference on Artificial Intelligence* (pp. 601–606). St. Paul, MN: AAAI Press.

Utgoff, P. E. (1989). Incremental induction of decision trees. *Machine Learning, 4*, 161–186.

Valiant, L. G. (1988). Functionality in neural networks. *Proceedings of the Seventh National Conference on Artificial Intelligence* (pp. 629–640). St. Paul, MN: AAAI Press.

Watanabe, L., & Rendell, L. (1990). Effective generalization of relational descriptions. *Proceedings of the Eighth National Conference on Artificial Intelligence* (pp. 875–881). Boston, MA: AAAI Press.

Weiss, S. M., & Kapouleas, I. (1989). An empirical comparison of pattern recognition, neural nets, and machine learning classification methods. *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence* (pp. 781–787). Detroit, MI: Morgan Kaufmann.

Widrow, B., & Hoff, M. (1960). Adaptive switching circuits. *IRE WESCON Convention Record, 4*.

Wilson, S. W. (1987). Classifier systems and the animat problem. *Machine Learning, 2*, 199–228.

Winston, P. H. (1984). Learning new principles from precedents and exercises. *Artificial Intelligence, 19*, 321–350.